Université Paris-Saclay

Rapport de Projet Machine Learning
M2 Maths de l'aléatoire: StatML

Modélisation de la location de vélos libre service : Open Bikes Challenge

---

Présenté par :

Issa-Mbenard DABO et Yassine HAMDI

issa-mbenard.dabo@ens-rennes.fr; yassine.hamdi@polytechnique.edu

---

Année Universitaire 2020-2021

Contents

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

Résumé

Dans ce travail, nous abordons l'analyse, la modélisation et la prévision de locations de vélos libres service du challenge Open Bikes 2016 à l'aide de diverses méthodes de régression. Nous avons mobilisé un large panel de modèles, méthodes et algorithmes statistiques et de machine learning afin d'expliquer et de prédire le flux de locations horaire de vélos libres service dans diverses villes et stations à l'aide de facteurs explicatifs en particulier météorologiques, convenablement choisis. Nous présentons dans ce travail les données étudiées, la démarche de modélisation utilisée et les principaux résultats obtenus. Les différentes analyses ont été réalisées avec le logiciel R (Comprehensive R Archive Network; CRAN) et un certain nombre de ses packages. Un package R rassemblant l'ensemble des codes mis en oeuvre et une interface graphique interactive permettant de visualiser le flux temporel des données de vélos libre service dans trois villes françaises ont été mis en place.

Chapter Un

Introduction

Les systèmes de mobilité durable, en particulier les vélos libre service sont de plus en plus courants dans la plupart des grandes villes du monde et offrent des solutions aux problèmes de la croissance des populations urbaines, l'augmentation des trafics automobiles, les embouteillages, la pollution de l'air et bien d'autres.
Ces dernières années ce type de mobilité a connu une très forte croissance dans toutes les grandes villes (Fishman, 2016), Paris, Lyon, Toulouse, New York, Londres, Barcelone, entre autres. De nouvelles stations de locations de vélos, de nouveaux systèmes plus innovants (par exemple électriques) voient régulièrement le jour entrainant une ré-organisation des voies de circulation dans les grandes villes. L'analyse, la gestion et la prévision des flux de ces moyens de transport sont ainsi devenues un enjeu majeur pour la planification urbaine, et la gestion des parcs de vélos. Ce domaine est au cœur des préoccupations de ce rapport.
En effet, nous nous sommes intéressés à l'analyse de données de transport urbain par vélo libres service dans certaines villes du monde à travers un pipeline de modèles et méthodes statistiques et de machine learning.
L'objectif principal de ce travail est de modéliser des déplacements futurs dans une station donnée. Nous nous sommes ainsi intéressés à l'étude de la variabilité de l'utilisation des vélos libres service au sein et entre les jours de semaine dans une ville donnée. Nous avons cherché à identifier des facteurs favorisant les déplacements à vélo, tels que les conditions météorologiques (pluie, température,...) temporels (week-end, jours ouvrés, vacances,...). La modélisation des déplacements par vélo libres service est de plus en plus abondante (Bouveyron et al., 2015; Borgnat et al., 2013; Etienne and Latifa, 2014; Faghih-Imani and Eluru, 2016; Froehlich et al., 2009; Saberi et al., 2018; Zhou, 2015; Gervini and Khanal, 2019; Han et al., 2018; Torti et al., 2021).
La presque majorité de cette littérature montre que l'utilisation des vélos libre service dépend des jours de la semaine et des heures de la journée (très prisé le matin, le soir, les jours ouvrés,...).
Nos analyses se concentrent sur les données de 30 stations de vélos de trois villes françaises (Lyon, Paris, Toulouse) entre avril et octobre 2016, provenant du Challenge Open Bikes 2016.
Dans un but de prévoir les déplacements futurs, nous avons d'abord exploré les données brutes dont le calendrier temporel était irrégulier (une ligne correspondant à une location à un temps donné) puis construit un calendrier temporel horaire pour chaque station. Ces données horaires ont été visualisées, analysées à l'aide de statistiques descriptives, de tests de comparaison (locations selon l'heure, le jour de la semaine, les conditions météorologiques,...). Nous avons ensuite fait concourir un ensemble de modèles statistiques et machine learning (modèle additif généralisé, réseaux de neurones, arbres de décision,....) afin de mettre en avant une meilleure prévision horaire des déplacements futurs (flux horaires de locations) en fonction de certains facteurs (conditions météorologiques,...). Nous avons développé un package R regroupant les codes des différentes analyses réalisées ainsi qu'une interface graphique interactive (à l'aide de RShiny) pour explorer les données des 30 stations considérées.
Le reste du document est organisé comme suit. Dans la Section 1.1 de ce chapitre, nous décrivons le cadre d'étude et le choix méthodologique proposé. Dans le deuxième chapitre, nous décrivons les méthodes de régression utilisées à un but prédictif des déplacements futurs et les résultats obtenus pour une station. Le document se termine par une conclusion et des annexes donnant les détails relatifs à la procédure de mise en œuvre des méthodes avec le logiciel R (version 4.0.3).

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

1.1 Cadre d'étude et choix méthodologiques

1.1.1 Présentation des données

Les données proviennent de la première partie du "Open Bikes 2016 Challenge" - une vue d'ensemble est donnée à ce lien. Celui-ci porte sur la prévision - en un sens précisé ci-dessous - du vélo-partage dans les villes de Lyon, Toulouse et Paris, à partir d'observations allant d'avril à octobre 2016. Il est constitué d'une part de données météo pour les trois villes d'étude, et d'autre part, pour chaque station de vélo-partage dans ces villes, d'un répertoire de tous les emprunts et retours de vélos. Un aperçu des différentes variables (date de location (moment), nombre vélos loués (bikes) disponibles (spaces), nébulosité (clouds, description), humidité (humidity), pression (pressure), température (temperature), vitesse du vent (wind)) et des paramètres statistiques descriptives sont exhibés dans les Tables 1.1-1.2 suivantes (voir la Table 1.3 pour plus de détail sur la description des noms des variables) sur les données météorologiques de la ville de Lyon ainsi que des données de vélo-partage pour une station dans cette ville (sur la place des Terreaux).

moment	bikes	spaces
2016-04-01 00:03:29	16	0
2016-04-01 00:13:01	15	1
2016-04-01 00:15:59	15	1
2016-04-01 00:16:49	14	2
2016-04-01 00:22:47	15	1
2016-04-01 00:24:09	15	1

bikes	spaces
Min. : 0.000	Min. : 0.000
1st Qu.: 2.000	1st Qu.: 2.000
Median : 8.000	Median : 8.000
Mean : 7.919	Mean : 7.758
3rd Qu.:14.000	3rd Qu.:13.000
Max. :16.000	Max. :16.000

Table 1.1: Vue et summary des données de location de la station Terreaux-Terme à Lyon

moment	clouds	description	humidity	pressure	temperature	wind
2016-04-01 00:00:00	90	light rain	71	1010	10.67	4.1
2016-04-01 01:01:26	92	light rain	81	1011	10.25	5.7
2016-04-01 01:18:56	92	light rain	81	1011	10.09	5.7
2016-04-01 01:39:57	92	light rain	87	1012	9.39	6.2
2016-04-01 01:59:47	90	light rain	87	1012	8.95	5.1
2016-04-01 02:20:34	90	light rain	87	1012	8.80	5.1

clouds	humidity	pressure	temperature	wind
Min. : 0.00	Min. : 16.0	Min. : 972.3	Min. :-1.29	Min. : 0.250
1st Qu.: 0.00	1st Qu.: 50.0	1st Qu.: 1013.0	1st Qu.:13.60	1st Qu.: 1.500
Median : 0.00	Median : 67.0	Median : 1017.0	Median :17.56	Median : 2.410
Mean : 27.12	Mean : 65.5	Mean : 1083.5	Mean :17.76	Mean : 2.864
3rd Qu.: 68.00	3rd Qu.: 81.0	3rd Qu.: 1020.0	3rd Qu.:21.99	3rd Qu.: 3.870
Max. :100.00	Max. :100.0	Max. :99385.0	Max. :33.78	Max. :21.590

Table 1.2: Vue et summary des données météo de Lyon

La plage d'observation est commune à toutes les stations : elle va du 1er avril 2016 au 5 octobre 2016 à 10h. Le challenge fixe un petit nombre de couples (nom de station, date) correspondant à début octobre, et pour lesquels il faut prédire le nombre de vélos présent dans la station au moment spécifié. La Figure 1.1 illustre à titre d'exemple la série des locations de vélo de la station Pomme, de Toulouse, Paris, respectivement, sur toute la période d'études et un zoom du 1 au 5 octobre 2016 en fonction des jours ouvrés et week-end (en rouge). Elle montre des flux différents en fonction des stations et du jour de la semaine. Ce constat est assez général au niveau des différentes stations. L'interface graphique interactive de visulisation des flux (horaires, journaliers, mensuels,...) des données des 30 stations sont disponibles via l'application Rshiny disponible à ce lien. La Figure 1.2 donne un aperçu de l'interface.

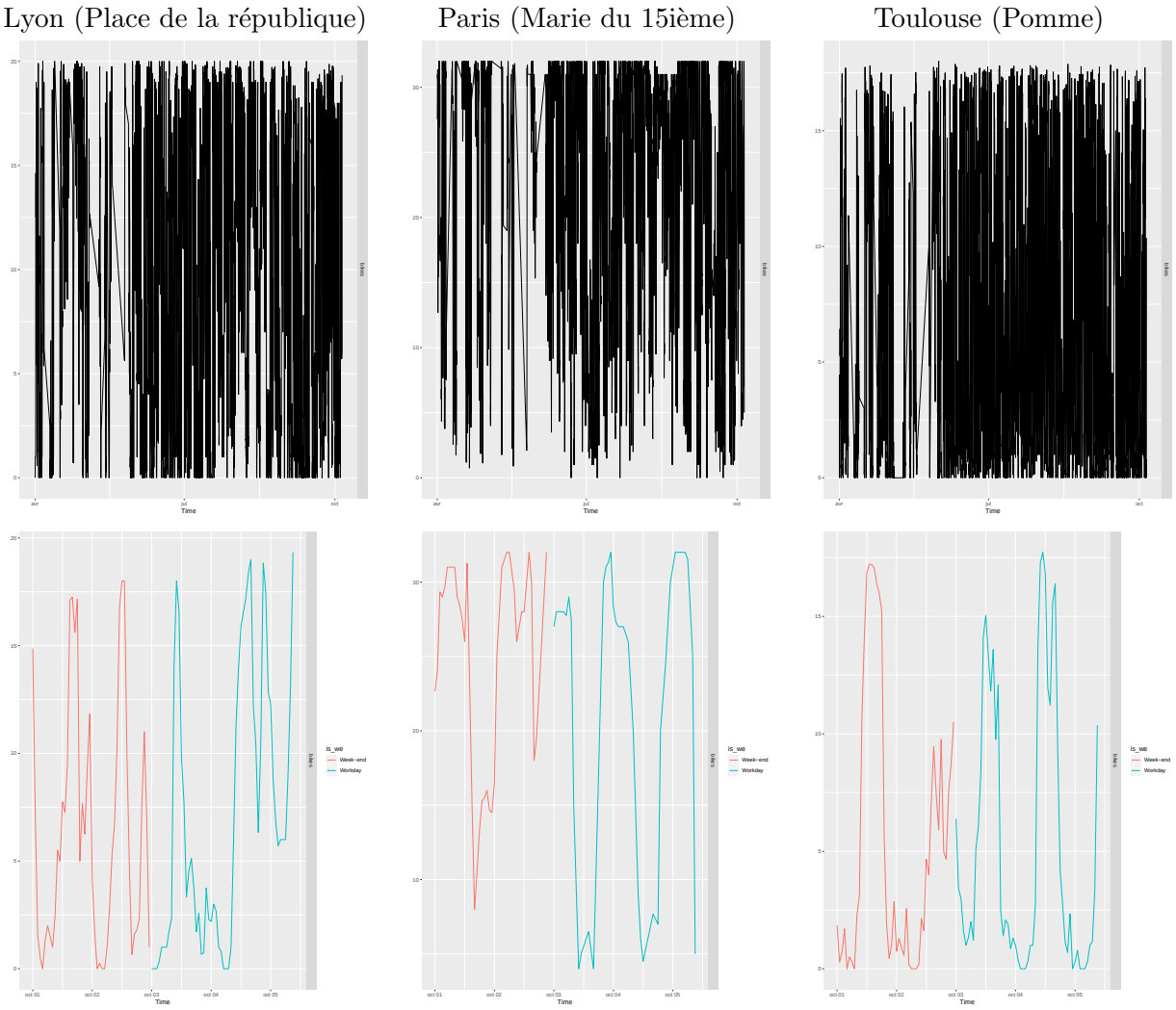

Figure 1.1: Locations de vélos d'avril à octobre et du 1 au 5 octobre 2016, des stations Place de la république, Marie du 15ième et Pomme de Lyon, Paris, Toulouse, respectivement

1.1.2 Deux méthodes

Lorsque nous avons tenté de prendre en compte une composante spatiale dans la loi des données, c'est-à-dire de regrouper les données des différentes stations en un seul ensemble d'apprentissage, nous nous sommes heurtés au problème du temps de calcul. Nous avons alors envisagé une seconde méthode : agréger les données par créneau de une heure. Néanmoins le problème du temps de calcul se posait toujours, si bien que nous avons décidé d'optimiser nos modèles sur une station à la fois. Les modèles que nous présentons dans la suite ont été sélectionnés en se basant sur les données de la station "Terreaux-Terme" à Lyon. Ceci ne nous empêche pas de nous demander, par curiosité, si nos modèles calibrés pour cette station ont des performances similaires sur d'autres stations.
Nous avons conservé les deux méthodes (données brutes ou agrégées), que nous avons explorées séparément. La seconde, avec des données agrégées par heure, semble également plutôt réaliste d'un point de vue applicatif puisque l'évolution des usages au cours d'une journée peut être fidèlement représentée par des données agrégées par heure ou par demi-heure, ce qui est fait dans un certain nombre de travaux récents dans le domaine, voir par exemple Torti et al. (2021).

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

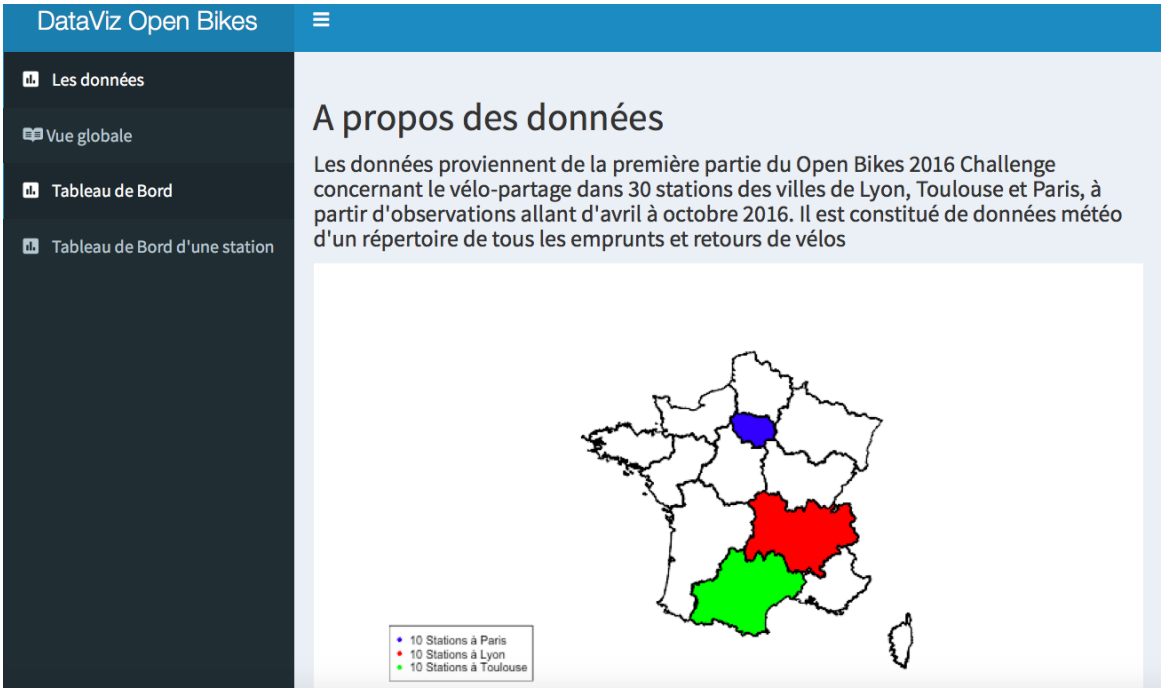

Figure 1.2: Interface interactive graphique Rshiny de visualisation des données

1.1.3 Approche séquentielle

Le but ultime de notre travail est la prédiction temporelle des partages de vélo (variable bikes) d'un jour donné d'une station donnée à l'aide des données des jours précédentes à la même station et de facteurs environnementaux de la station.
La liste de couples (nom de station, datetime) de l'échantillon de test proposée par le challenge ne comporte que 10 datetime différents répartis sur 5 jours : du 5 octobre 2016 au 9 octobre 2016. Il est naturel que les prédictions portant sur le cinquième jour soient moins bonnes que sur le premier.Afin de voir si nous pouvions obtenir une grande précision sur nos prédictions, nous avons choisi de nous détacher de l'approche du challenge et de nous attaquer au problème consistant à prédire un jour en ayant observé les jours précédents. Autrement dit nous prédisons ce qui se passera au jour J à l'aide des données des jours précédents, puis pour prédire le jour $J+1$, nous prenons les observations du jour J, et ensuite prédisons ce qui se passera au jour $J+1$, et ainsi de suite. Nous pensons que cette approche séquentielle correspond au contexte applicatif réaliste suivant. Durant une journée, certaines stations se vident naturellement de leurs vélos, mais l'entreprise peut répondre aux besoins des usagers, malgré les contraintes de taille des stations, en déplaçant des vélos de stations peu fréquentées à un certain moment de la journée vers des stations où la demande sera plus importante à ce moment. Pour la planification au jour le jour de ces transferts, l'entreprise peut bel et bien s'appuyer sur les données de la veille. L'approche que nous avons choisie vise à répondre à cette problématique opérationnelle. L'approche proposée par le challenge semble correspondre aux autres problèmes très réalistes d'anticiper des changements d'usage, notamment à l'approche de jours fériés ou spéciaux, et pour le travail prospectif de changement de stratégie de répartition de vélos ou de définition d'une stratégie d'investissement, ...
Afin de mettre en œuvre notre approche, nous avons choisi de prendre comme données de test le dernier mois du jeu de données, c'est-à-dire du 6 septembre au 5 octobre 2016 inclus. Nous entrainons donc nos algorithmes sur la période du 1er avril 2016 au 5 septembre 2016 inclus.
Nous avons également considéré une approche séquentielle avec ré-entrainement de toutes les heures pour le jeu de données agrégées par heure.

1.1.4 Méthode de validation croisée

Tout d'abord, notre fonction de perte est la perte quadratique - précisons que nous faisons en sorte que tous nos prédictions soient positives ou nulles, donc nous passons à la partie positive avant de prendre la perte quadratique. Tout au long de notre démarche, pour choisir les différents paramètres de nos modèles, nous avons utilisé une validation croisée par blocs. Nous savons que son comportement théorique, en termes de sélection de modèles, est bien étudié -et les performances généralement satisfaisantes- dans le cadre de données i.i.d., ce qui ne correspond pas à notre situation. Nous sommes conscients qu'il existe des méthodes de validation croisée généralement plus adaptées aux données temporelles. Mais celles que nous avons trouvées (Time series split & Blocked cross-validation) consistent -au moins en partie- à entraîner sur de petites portions de l'ensemble d'entraînement, ce qui, nous semble-t-il, pourrait entraîner un biais puisque notre but est d'approcher la sélection de modèles optimale parmi nos modèles lorsqu'ils sont entraînés sur tout le jeu de données. La petite taille de notre jeu de données (5 mois) relativement à la périodicité du phénomène étudié (de l'ordre de la semaine) pourrait éventuellement participer à la création de ce biais. Par manque de temps, nous avons choisi de ne pas nous plonger dans la littérature des analyses théoriques et expérimentales de ces différentes méthodes, et de nous reposer sur le principe heuristique suivant : nous avons vu en cours qu'on peut néanmoins obtenir une bonne sélection de modèles dès lors que les blocs sont suffisamment grands pour que la corrélation entre des points de données de deux blocs différents situés relativement loin des bords des blocs soit faible. Comme on peut le voir sur le diagramme d'auto-corrélation de la Figure 2.13, où une semaine correspond à un lag d'environ 1500, une taille de blocs d'un peu plus de 2 semaines (lag de 4000 environ) semble être un bon compromis, afin de ne pas avoir trop peu de blocs. Nous avons retenu une cross-validation 8-fold.

```
> acf(data_station$bikes, lag.max = 10000)
```

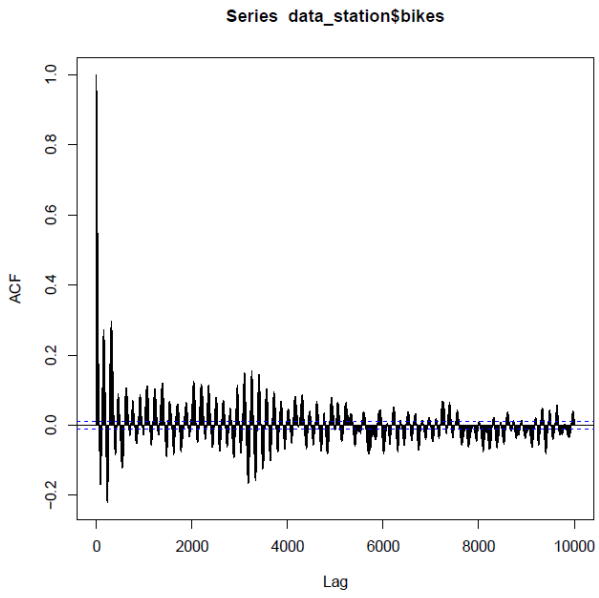

Figure 1.3: diagramme d'auto-corrélation des données

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

1.1.5 Traitement préliminaire, création de premières features

1.1.5.1 Fusion des données de disponibilités de vélos avec les données météo

Comme le nombre de vélos disponibles n'est relevé que lorsqu'un vélo est emprunté ou rendu, les instants correspondants ne sont naturellement pas identiques à ceux des données météo. Nous avons d'abord essayé de créer une colonne comprenant un arrondi du datetime à la demi-heure, puis d'effectuer une jointure des tables sur cette colonne. Malheureusement, des incompatibilités subsistaient. C'est pourquoi nous avons finalement codé un parcours des tables afin d'attribuer à chaque instant d'emprunt/retour de vélo l'instant le plus proche dans les données météo.

1.1.5.2 Jour précédent et données manquantes

Comme les usages sur deux jours ouvrés consécutifs sont souvent très proches, notamment dans le cas de faible variation des conditions météorologiques, nous avons souhaité associer à chaque instant d'emprunt/retour le nombre de vélos dans la station une heure avant et 24 heures avant, créant ainsi les variables `bikesPrec` et `bikesPrec7`. Nous nous sommes heurtés aux problèmes des données manquantes : il existe certaines dates dans la plage d'étude (avril-septembre 2016) pour lesquelles il n'y a aucune donnée. Diverses méthodes s'offraient à nous, dont l'utilisation d'un modèle d'apprentissage basé sur les données disponibles, ou la complétion d'une matrice de faible rang (en supposant que les données quotidiennes soient proches d'un espace de faible dimension). Malheureusement, par manque de temps, nous n'avons pas exploré ces pistes. Nous nous sommes donc limités à une solution temporaire consistant à chercher, par un parcours des tables, pour chaque instant d'emprunt/retour, l'instant le plus proche soit dans le jour précédent, soit dans le jour le plus proche pour lequel des données sont disponibles. Nous verrons par la suite que cette variable se révèle néanmoins pertinente.

1.1.5.3 Features temporelles

Comme vu en Section 1.1.1, les informations temporelles dans les données brutes sont au format datetime. Ce format n'étant pas directement utilisation dans différentes méthodes d'apprentissage, nous avons commencé par le transformer en nombre réel à partir de l'écart de la date à celle du premier jour étudié. Deux choix différents d'unité pour la conversion donne les variables `month` (nombre réel entre 0 et 12) et `time`. Ensuite, comme il est naturel que l'heure de la journée et le jour de la semaine jouent ont une grande influence sur la quantité de vélos disponibles, nous avons créé deux variables `hour` et `dow` (pour "day of the week"), respectivement un nombre réel entre 0 et 24 et un entier entre 1 et 7. Nous avons aussi rajouté une variable catégorielle `cat_month` (pour "categorial month") pour le numéro de mois, une variable `seasons` entre 1 et 4 pour les saisons et une variable booléenne `holidays` pour les vacances scolaires et les ponts du mois de mai. Après traitement des données, on obtient les variables explicatives décrites dans le Tableau 1.3.

1.1.6 Analyses descriptives des données

Nous avons réalisé une analyse descriptive des données afin d'explorer d'éventuelles relations entre la variable étudiée (bikes) et les facteurs explicatifs. Nous avons ainsi calculé la matrice de corrélation linéaire des données brutes quantitatives et comparé la distribution des bikes en fonction des modalités des facteurs explicatifs qualitatifs.
On remarque ainsi (Figure 1.4) que `bikes` est légèrement corrélé négativement avec `temperature`.

`moment`	Date et heure de l'observation
`clouds`	Quantité de nuage dans le ciel
`humidity`	Humidité dans l'air
`temperature`	Température en °C
`wind`	Force du vent
`pressure`	Pression de l'air en hPa
`description`	Description de la météo
`date`	Date de l'obersvation
`month`	Mois
`time`	Valeur numérique de la variable `moment`
`hour`	Heure
`BikesPrec`	Nombre de vélos dans la station à l'heure précédente
`dow`	Jour de la semaine
`cat_month`	Mois
`seasons`	Saison
`holidays`	Variable précisant si on est en période de vacances ou non
`is _ we`	Variable précisant si on est en Week-end ou non
`bikesPrec7`	Nombre de vélos dans la station 24h auparavant

Table 1.3: Noms des variables

Cela est en accord avec l'intuition naturelle selon laquelle les usagers préfèrent louer des vélos quand il fait chaud. Cependant le coefficient de corrélation linéaire n'étant pas très fort, on peut suspecter qu'une relation linéaire enter ces deux variables ne serait pas judicieuse. On remarque également que `humidity` est corrélée positivement avec `clouds`, ce qui paraît logique car quand il pleut, il y a forcément des nuages. On a aussi `temperature` corrélée négativement avec `humidity`, indiquant qu'il fait souvent plus froid quand il pleut. On note une corrélation linéaire positive pas très conséquente entre `temperature` et `hour` ou `month`, autrement dit l'utilisation des vélos libre service entre le printemps et l'été (où les jours sont longs et la température élevée) augmenterait pendant ces mois.
Différentes boîtes à moustaches combinées à des tests non-paramétriques (de Kruskal-Wallis, ne supposant pas de loi sur la distribution de la variable étudiée) de comparaison des moyenne de `bikes` dans les différentes villes, selon le jour de la semaine, les périodes de vacances, ont été réalisés (Figures 1.5-1.7).
On remarque ainsi qu'il y a en générale plus de vélos dans les stations parisiennes que dans les autres villes et qu'il y en a généralement plus à Lyon qu'à Toulouse, ce qui n'est pas surprenant au regard de la population de ces villes, Paris est la plus peuplée (20755 hab/km^2 de densité de population, sondage 2017) des trois villes suivie de Lyon (10781 hab/km^2 de densité de population, sondage 2017) et Toulouse (4053 hab/km^2 de densité de population, sondage 2017).
On remarque dans la Figure 1.6, qu'il y a en moyenne moins de vélos loués le week-end à Paris et à Toulouse alors que le comportement des vélos semble stable toute la semaine à Lyon. On remarque (Figure 1.7), qu'il y a en moyenne moins de vélos loués pendant les vacances à Lyon et à Toulouse qu'à Paris. Dans le sud, les vacances favoriseraient l'augmentation de l'utilisation de ce transport urbain.
Les séries de locations (agrégées) de voiture dans les 10 stations de Paris, suivant divers horizons temporels ont été représentées dans les Figures 1.8-1.10. Dans la Figure 1.8, chaque courbe représente une station. Les graphiques des autres stations sont disponibles au niveau de l'interface graphique Rshiny. On remarque ainsi une certaine périodicité des données. Chaque mois il y a des pics qui apparaissent à peu près aux mêmes endroits, ce qui pourrait faire penser à un schéma dans l'évolution du nombre de vélos dans une station. Cependant, il est à noter un comportement différent selon le mois, entre avril et juillet. Dans la Figure 1.9, chaque couleur représente une semaine du mois d'avril. Dans toutes les stations et pour chaque semaine, on peut distinguer deux comportements différents. En début de semaine, on note un schéma plus ou moins cyclique, un cycle semble correspondre à un jour. Le week-end, l'évolution du nombre de vélo paraît plus erratique.

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

Les courbes de la Figure 1.10 représentent les jours du mois d'avril. Dans chaque station, le même schéma semble se répéter tous les jours, qui pourrait correspondre au cycle évoqué précédemment. Par exemple dans la première station *(Concorde)*, on remarque qu'il y a souvent très peu de vélos disponibles entre 21h et 06h mais contrairement à la période 09h-18h. Par contre ce schéma ne semble pas être unique, on peut remarquer le comportement de la septième station (*Dugommier*) est très différent de *(Concorde)*. Les représentations graphiques des series temporelles étudiées (de pas de temps irrégulier ou horaire) ont été réalisées à l'aide du package R fpp3 et le livre de Hyndman and Athanasopoulos (2018).

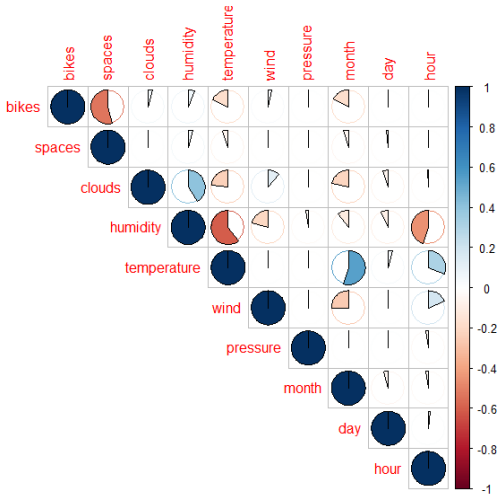

Figure 1.4: Corrélation linéaire entre les variables explicatives quantitatives

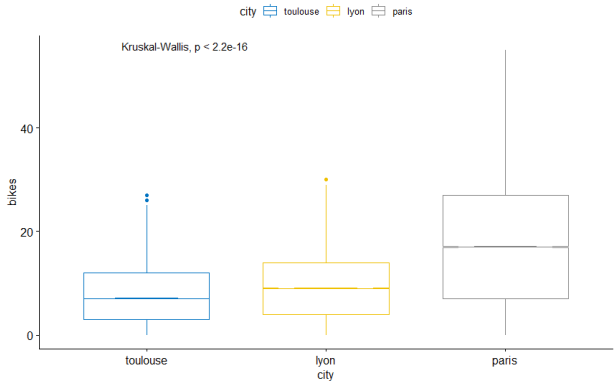

Figure 1.5: Boxplot du nombre de vélos dans chaque ville

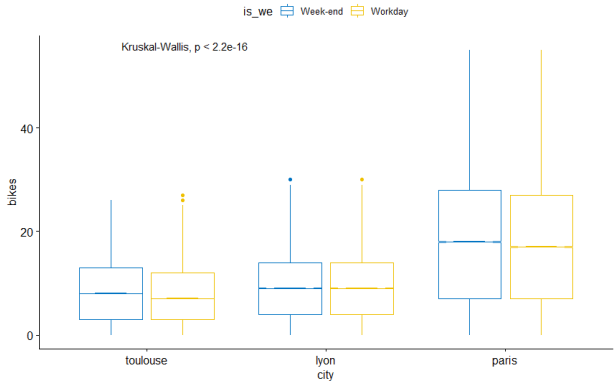

Figure 1.6: Boxplot du nombre de vélos dans chaque ville en semaine et en week-end

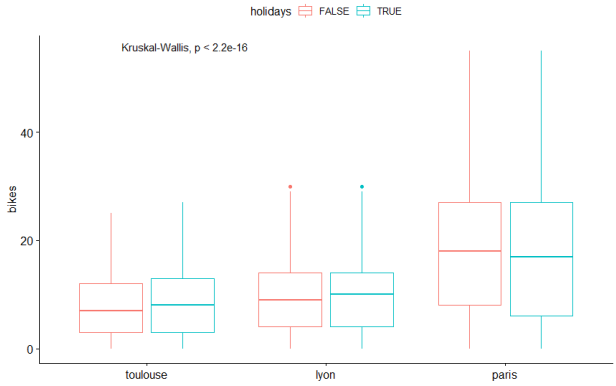

Figure 1.7: Boxplot du nombre de vélos dans chaque ville en période de vacances ou non

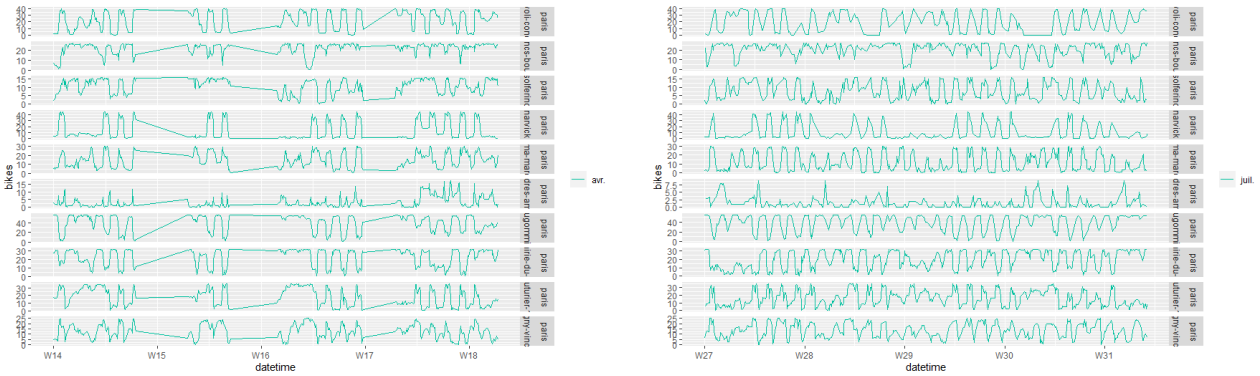

Figure 1.8: Evolution du nombre de vélos dans 10 stations de Paris pour le mois d'avril (à gauche) et le mois de juillet (à droite)

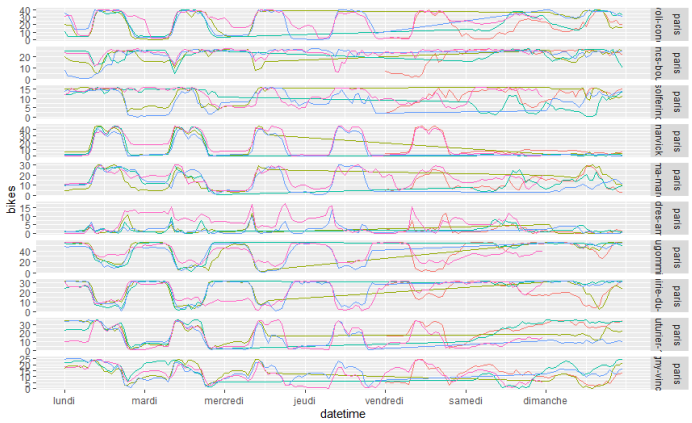

Figure 1.9: Evolution du nombre de vélos dans 10 stations de Paris par semaine

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

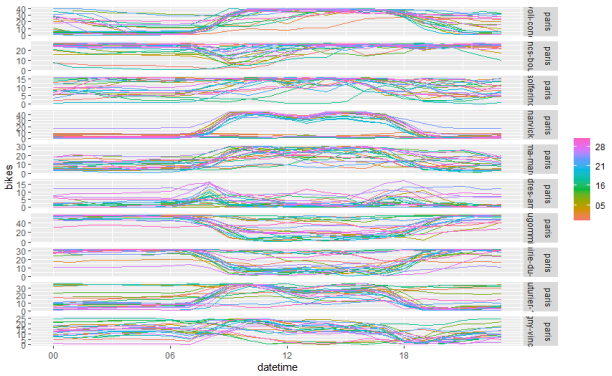

Figure 1.10: Evolution du nombre de vélos dans 10 stations de Paris par jour

Chapter Deux

Modélisation

Dans ce chapitre nous donnons la démarche de modélisation des données de vélo-partage dans une station donnée. Pour ce faire, on introduit $(X_{t_{i,j}}, Y_{t_{i,j}}, Z_{t_{i,j}}) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^p$, avec $j = 1, ..., n$, $i = 1, ..., n_j$, $t_{i,j} \in \{0, 1, ..., 23\}$, j étant le numéro du $j$ième jour de la période du 1 avril 2016 au 5 octobre 2016, $n = 167$ le nombre de jours de cette période, n_j le nombre de temps d'observation des données au $j$ième jour. Supposons que $X_{t_{i,j}}, Y_{t_{i,j}}, Z_{t_{i,j}}$ sont des variables aléatoires considérées comme observations temporelles du vecteur X (représentant les variables explicatives quantitatives des données étudiées), Z (les variables explicatives qualitatives), et Y (la variable mesurant le nombre de locations de vélos) de 0 à 23h, du 1 avril 2016 au 5 octobre 2016 (voir la Table 1.3 pour plus de détails sur ces variables).
Comme annoncé dans le chapitre précédent, nous considérons deux cadres temporels dans la suite, le cas des données brutes initiales où les temps d'observation sont continus, irréguliers ($t_{i,j}[0, 24]$), le cadre horaire (données agrégées par heure; $t_{i,j} \in \{0, 1, ..., 23\}$).

2.1 GAM

Nous nous sommes, tout d'abord intéressés aux modèles additifs. Nous rappelons ici succinctement le modèle additif généralisé GAM (Generalize Additive Model) nous renvoyons vers (Hastie and Tibshirani, 1990; Wood et al., 2015; Fasiolo et al., 2020; Li and Wood, 2020), pour plus de généralités.

Soit L une loi faisant partie de la famille exponentielle. On notera μ son espérance et θ son paramètre naturel. Selon(Hastie and Tibshirani, 1990), dans certains cas on appelle "lien canonique" la fonction g telle que $g(\mu) = \theta$. Un modèle GAM consiste à supposer que pour tout (z, x), la loi conditionnelle $P_{Y_{t_{i,j}}|(Z_{t_{i,j}}, X_{t_{i,j}})=(z,x)}$ est une loi L telle que

$$\tilde{g}(\mu(z, x)) = \beta_0 + \sum_k \beta_k z_k + \sum_l s_l(x_l),$$

pour une certaine fonction $\tilde{g}$, où les β_k sont des constantes à estimer, $s_l = \sum_{i=1}^{k_l} \alpha_{l,i}\varphi_i$ est une fonction spline de degré k_l, φ_i un élément d'une base de régression, $\alpha_{l,i}$ des coefficients à estimer. Dans le cas où $\tilde{g} = g$, le paramètre naturel de $P_{Y_{t_{i,j}}|(Z_{t_{i,j}}, X_{t_{i,j}})=(z,x)}$ est:

$$\theta(z, x) = \beta_0 + \sum_k \beta_k z_k + \sum_l s_l(x_l).$$

Par exemple, pour la loi de Poisson, le lien canonique est le logarithme népérien (et le paramètre naturel est $\theta = \log(\lambda)$). Selon (Hastie and Tibshirani, 1990), pour la loi normale, le lien canonique est la fonction identité. On suppose de plus, si l'on note $Y_{t_{i,j}} = \mu(Z_{t_{i,j}}, X_{t_{i,j}}) + \varepsilon_{i,j}$, que les $\varepsilon_{i,j}$ sont des variables aléatoires réelles i.i.d.
Comme énoncé précédemment, dans toute la suite nous nous sommes concentrés sur une seule station

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

(Lyon, place des Terreaux) afin de trouver un modèle de prévision adapté. Nous avons entraîné nos modèles sur la période Avril-Août 2016 et toute nos prédictions ont été faites sur le mois de Septembre et la première semaine d'Octobre. Il en sera de même pour toutes les méthodes utilisées.

2.1.1 Approche naïve

Nous avons utilisé le package `mgcv` et sa fonction `gam` (Wood and Wood, 2015) pour cette partie du projet. Nous avons abordé, en premier lieu le problème par une approche naïve pour pouvoir nous approprier le package. Nous avons établi un modèle additif (sous l'hypothèse gaussienne) avec quelques splines basiques en ne considérant que peu de variables et sans spécifier de degrés comme vous pouvez le voir dans le code ci-dessous concernant les données temporelles initiales.

```
> data = read.csv("data_rapport.csv")
> equation = bikes ~ factor(is_we)+factor(seasons)+factor(holidays)+ factor(dow
+ )+s(hour)+s(month) +s(wind) +s(humidity)+s(temperature)+s(pressure)
> g = gam(equation,data = data)
```

On considère alors le modèle additif suivant sur les données brutes non agrégées :

$$\hat{Y} = \hat{g}(X,Z) = \hat{\beta}_0 + \hat{\beta}_1 * \texttt{is\_we} + \hat{\beta}_2 * \texttt{seasons} + \hat{\beta}_3 * \texttt{holidays} + \hat{\beta}_4 * \texttt{dow} + \hat{s}_1(\texttt{hour}) + \hat{s}_2(\texttt{month}) \quad (2.1)$$

$$+\hat{s}_3(\texttt{wind}) + \hat{s}_4(\texttt{humidity}) + \hat{s}_5(\texttt{temperature}) + \hat{s}_6(\texttt{pressure}).$$

L'estimation du modèle est donné dans la Figure 2.1. Pour évaluer les performances des modèles gam

```
Family: gaussian
Link function: identity

Formula:
bikes ~ factor(is_we) + factor(seasons) + factor(holidays) +
    factor(dow) + s(hour) + s(month) + s(wind) + s(humidity) +
    s(temperature) + s(pressure)

Parametric coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           6.10362    0.10603  57.562  < 2e-16 ***
factor(is_we)Workday  2.22993    0.05739  38.858  < 2e-16 ***
factor(seasons)2     -1.95052    0.22170  -8.798  < 2e-16 ***
factor(seasons)3     -0.45422    0.32898  -1.381  0.16739
factor(holidays)TRUE -0.30047    0.10115  -2.971  0.00297 **
factor(dow)2          3.05579    0.07778  39.286  < 2e-16 ***
factor(dow)3          2.41337    0.08436  28.609  < 2e-16 ***
factor(dow)4          1.78597    0.08876  20.120  < 2e-16 ***
factor(dow)5          2.25061    0.06286  35.803  < 2e-16 ***
factor(dow)6          1.76861    0.05864  30.161  < 2e-16 ***
factor(dow)7         -0.14554    0.05840  -2.492  0.01270 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                edf Ref.df       F p-value
s(hour)       8.970  9.000 3242.81  <2e-16 ***
s(month)      8.762  8.983  112.33  <2e-16 ***
s(wind)       8.834  8.991   31.42  <2e-16 ***
s(humidity)   8.897  8.995   21.50  <2e-16 ***
s(temperature) 8.805  8.989   14.80  <2e-16 ***
s(pressure)   8.962  8.999  111.53  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Rank: 64/65
R-sq.(adj) =  0.571   Deviance explained = 57.2%
GCV = 14.201  Scale est. = 14.171    n = 29827
```

Figure 2.1: Résumé du modèle

nous avons utilisé le R^2 et le `rmse` (moyenne des carrés des erreurs). Dans le cas de l'approche naïve on peut voir qu'aucun de ces deux indicateurs n'est vraiment satisfaisant. Nous avons alors obtenu un R^2 de 0.5 environ et un `rmse` de 5.6 de test, le modèle n'est donc pas encore assez précis, il faut l'améliorer.
Nous avons ensuite remarqué que l'erreur de prédiction résultante du modèle varie en fonction des degrés, des bases de splines choisies et de la famille choisie.
Nous avons d'abord pensé à utiliser une loi positive pour $\varepsilon_{i,j}$, car la variable à prédire prend des valeurs positives. Nous avons donc considérer la loi de Poisson mais cela empire la performance du premier modèle (modèle de l'équation (2.1)). Nous avons donc finalement décidé de conserver la loi par défaut, loi gaussienne, et si une valeur négative est prédite, nous la mettons à 0.
Pour améliorer la performance de ce premier modèle gam, nous allons considérer un meilleur choix des bases de splines `cr`, nous verrons plus tard comment choisir les degrés.

2.1.2 Cart

Lors de nos travaux, nous nous sommes intéressés aux arbres de régression (Breiman et al., 1984), grâce au package `rpart`. Cette méthode produit une prédiction à partir d'un arbre de décision dont chaque branche représente une condition sur une variable explicative. Malgré le fait qu'ils se soient montrés plutôt inefficaces pour la prédiction, ils nous ont été utiles pour élaborer le choix des paramètres et variables dans le modèle additif. Dans un premier temps, les arbres nous ont permis de créer de nouvelles variables.

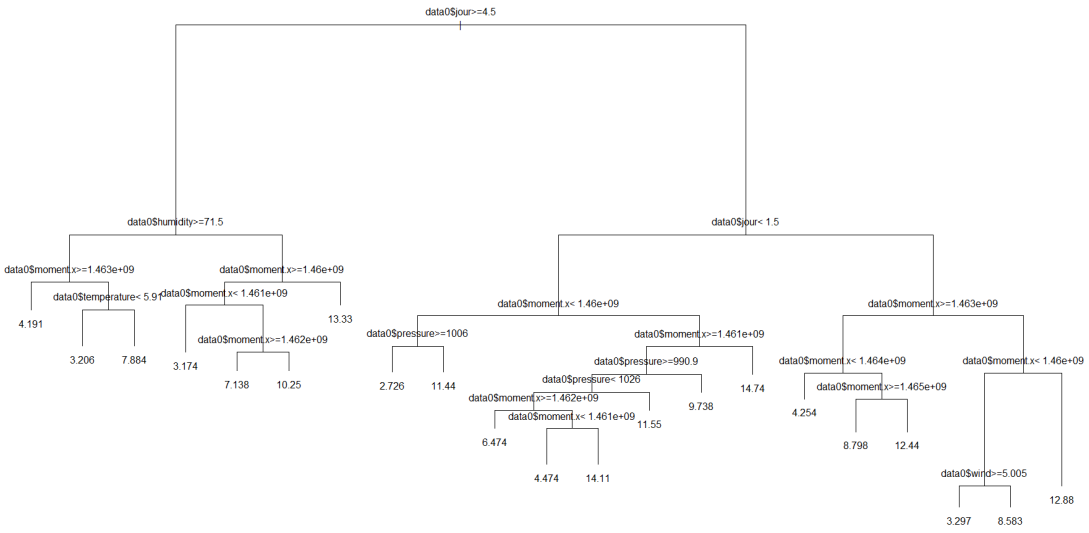

Figure 2.2: Arbre de régression réalisé à partir des données de la station

Dans l'arbre de la Figure 2.2, la racine sépare les 4 premiers jours de la semaine des autres, nous avons donc réalisé que la valeur que l'on veut prédire se comporte différemment le week-end et les jours ouvrés, ce qui confirme ce que nous avions remarqué dans la partie descriptive. Nous avons donc créé une variable qui indique si on est en week-end ou non. Nous avons été emmenés à créer d'autres variables de cette manière, dont une pour la forte humidité par exemple, mais celles-ci n'améliorant pas la modélisation, nous ne les avons donc pas considérées.
Nous avons ensuite voulu trouver les splines pertinents pour la modélisation par modèle additif. Pour cela, nous avons créé un arbre de régression avec toutes les variables. Nous l'avons ensuite élagué pour supprimer toutes les branches superflues. On obtient alors l'arbre présenté dans la Figure 2.3.

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

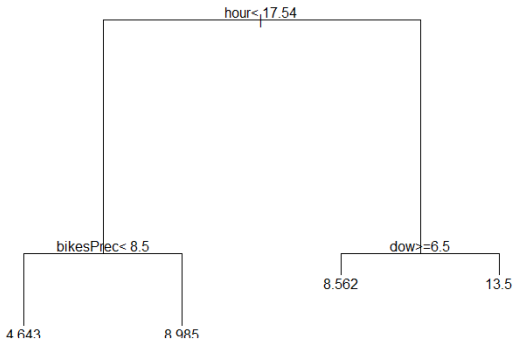

Figure 2.3: Arbre de régression élagué

Les noeuds reliés dans l'arbre permettent de trouver les relations amenant à considérer de nouveaux splines pour la modélisation additive généralisée. Les branches de l'arbre nous permettent d'exhiber des relations entre les variables que nous utiliserons pour créer les splines du modèle `gam`. On remarque dans cet arbre, qu'une branche relie les variables **hour** (l'heure de la location) et **bikesPrec** (nombre de vélos dans la station l'heure précédente la location), on va donc utiliser un spline à partir de ces deux variables, et on procède de même pour trouver d'autres relations.
On remarque que peu de variables apparaissent dans l'arbre, on ne peut donc pas trouver beaucoup de relations. Nous avons alors créé d'autres arbres dans lesquels au moins une de ces deux variables ne figure pas pour laisser plus de place aux autres variables et ainsi trouver plus de relations. Nous avons ainsi enlevé la variable **bikesPrec** dans l'arbre pour créer un nouvel arbre et nous procèderons de la sorte jusqu'à avoir assez de relations.
A l'aide de ces relations, nous avons obtenu un nouveau modèle additif où il a fallu préciser des degrés pour chaque spline, le code suivant illustre le modèle estimé :

```
bikes ~ bikesPrec + s(hour, k = 10, bs = "cr") + s(hour, dow,
    k = 37) + s(hour, bikesPrec, k = 41) + s(month, k = 3, bs = "cr") +
    s(dow, k = 7, bs = "cr") + s(humidity, k = 3, bs = "cr") +
    s(temperature, k = 4, bs = "cr") + s(pressure, k = 3, bs = "cr") +
    s(month, humidity, k = 18) + s(month, temperature, k = 14) +
    s(month, pressure, k = 13)
```

On a ainsi obtenu le modèle additif suivant sur les données brutes :

$$\hat{Y} = \hat{g}(X,Z) = \hat{\beta}_0 + \hat{\beta}_1 * \texttt{bikesPrec} + \hat{s}_1(\texttt{hour}) + \hat{s}_2(\texttt{hour},\texttt{dow}) + \hat{s}_3(\texttt{bikesPrec},\texttt{hour}) + \hat{s}_4(\texttt{month}) + \hat{s}_5(\texttt{dow}) \tag{2.2}$$
$$+ \hat{s}_6(\texttt{humidity}) + \hat{s}_7(\texttt{temperature}) + \hat{s}_8(\texttt{pressure}) + \hat{s}_9(\texttt{humidity}, + \hat{s}_{10}(\texttt{month})$$
$$+ \hat{s}_{11}(\texttt{month},\texttt{temperature}) + \hat{s}_{12}(\texttt{month},\texttt{pressure}).$$

2.1.3 Degrés

Pour chaque spline du modèle, il faut choisir le degré. Nous avons calibré ce paramètre par validation croisée (conférer la Figure 2.4). L'analyse de cette figure qui donne l'erreur de prévision d'un spline en fonction du degré permet de remarquer que le `rmse` n'évolue plus à partir de $k = 9$ (degrès), on choisit donc cette valeur pour le degré de ce spline. On procède de manière similaire pour tous les

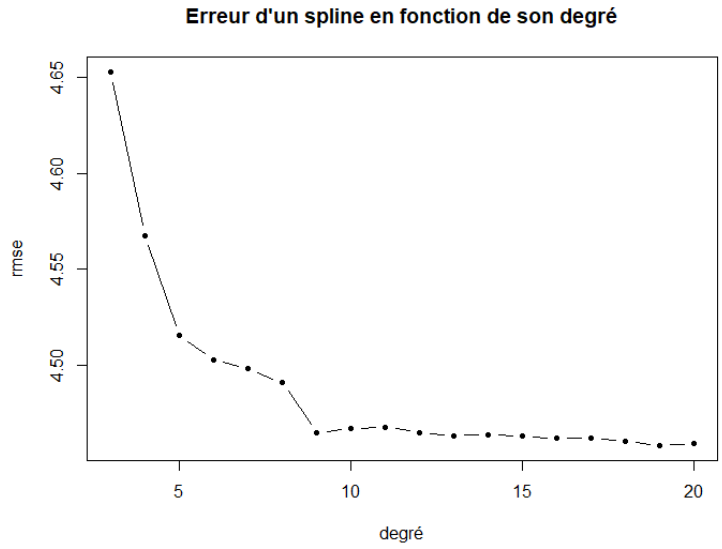

Figure 2.4: Erreur de validation croisée d'un spline en fonction de son degré

autres splines.
Et on obtient ainsi le modèle suivant sur les données brutes :

$$\hat{Y} = \hat{g}(X,Z) = \hat{\beta}_0 + \hat{\beta}_1 * \texttt{bikesPrec} + \hat{s}_1(\texttt{hour}, k_1 = 10) + \hat{s}_2(\texttt{hour}, \texttt{dow}, k_2 = 37) \tag{2.3}$$
$$+\hat{s}_3(\texttt{bikesPrec}, \texttt{hour}, k_3 = 41) + \hat{s}_4(\texttt{month}, k_4 = 3) + \hat{s}_5(\texttt{dow}, k_5 = 7) + \hat{s}_6(\texttt{humidity}, k_6 = 3)$$
$$+\hat{s}_7(\texttt{temperature}, k_7 = 4) + \hat{s}_8(\texttt{pressure}, k_8 = 3) + \hat{s}_9(\texttt{humidity}, \texttt{month}, k_9 = 18)$$
$$+\hat{s}_{10}(\texttt{month}, \texttt{temperature}, k_{10} = 14) + \hat{s}_{11}(\texttt{month}, \texttt{pressure}, k_{11} = 13).$$

Ce modèle **gam** (2.3) obtenu par validation croisé du degrés de liberté montre une légère amélioration, on passe d'un `rmse` de test4 à 4.2. L'analyse de l'auto-corrélation dans les résidus de ce modèle montre une présence d'autocorrelation, comme on peut le voir dans la Figure 2.5, on va donc essayer d'y remédier.

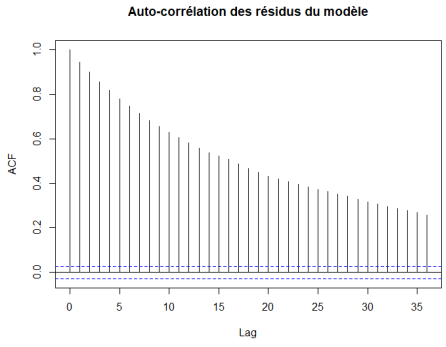

Figure 2.5: Auto-corrélation des résidus de la prédiction du modèle (2.3)

2.1.4 Auto-corrélation dans les résidus

Afin de remédier à la corrélation dans les résidus du modèle (2.3), nous devons ajuster un modèle adapté aux résidus. Nous avons ainsi modélisé les résidus d'abord avec les forêts aléatoires mais cela n'a

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

pas été concluant. Nous avons ensuite utilisé les modèles de séries temporelles (AR, ARMA, ARIMA) qui ont donné de meilleurs résultats. Cette modélisation des résidus a été un point déterminant dans le choix de l'agrégation des données par heure. En effet, dans les données initiales brutes, la durée séparant deux observations est très aléatoire, ce qui nous rendait difficile l'ajustement à un modèle type ARIMA. Après avoir refait le choix du lissage par splines sur les données agrégées horaires, nous avons ensuite utilisé la fonction `auto.arima` du package `forecast` afin de trouver automatiquement le meilleur modèle sur les résidus `ARIMA` pour $\hat{\varepsilon} = Y - \hat{Y}$. Le meilleur modèle proposé est un `ARIMA(1,0,2)`. L'auto-corrélation des résidus devient alors négligeable (Figure 2.6).

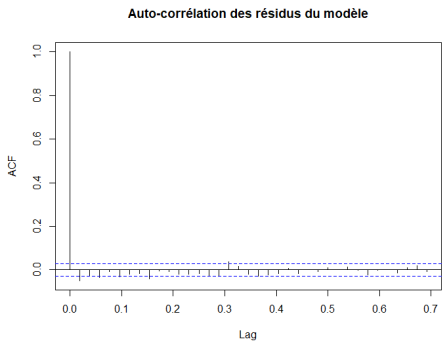

Figure 2.6: Auto-corrélation des résidus de la prédiction du gam

En utilisant ce modèle `ARIMA(1,0,2)` sur les résidus et le modèle `gam` (2.3) avec les données agrégées, nous obtenons une nouvelle prédiction pour le nombre de vélos via le modèle de prévision suivant :

$$\hat{Y}_{t+1} = \hat{g}(\mathbf{X}, \mathbf{Z}) + f(\hat{\varepsilon}), \tag{2.4}$$

où $\hat{g}(\mathbf{X}, \mathbf{Z})$ est l'ajustement basé sur le modèle `gam` (2.3) avec les observations jusqu'au temps t et $f(\hat{\varepsilon})$ est l'ajustement du résidu par un modèle `ARIMA(1,0,2)`.
En plus d'éliminer l'auto-corrélation, nous avons amélioré le modèle type (2.3) avec auto-corrélation des résidus, avec un R^2 de 0.64 et un `rmse` de 2.3 sur les données agrégées par heure et 1.6 sur les données brutes, l'amélioration est assez significative par rapport aux modèles précédents.
La prédiction sur les premiers jours d'octobre via le modèle (2.4) est donné à la Figure 2.7 qui permet une prediction assez fidèle à la réalité mais encore améliorable.

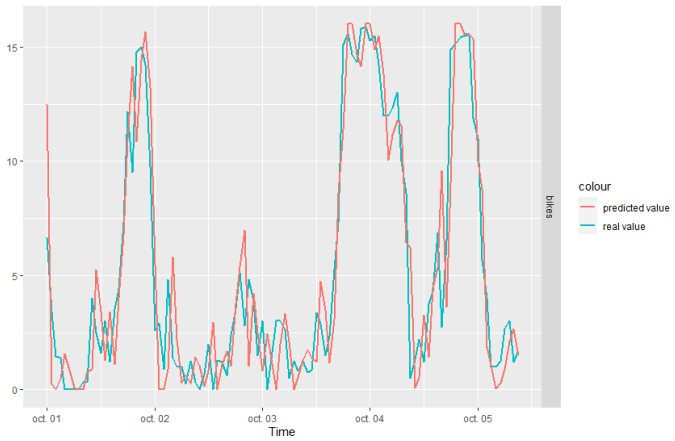

Figure 2.7: Comparaison entre valeur réelle et prédiction par le modèle (2.4) issu d'un gam avec modélisation de l'auto-corrélation des résidus, pour la première semaine d'Octobre

2.2 Méthodes d'ensemble

Nous allons présenter deux nouvelles approches basées sur des méthodes d'ensemble que nous avons explorées. La première repose sur l'utilisation de la méthode de gradient boosting et la deuxième utilise les forêts aléatoires.

2.2.1 Gradient boosting

Nous avons utilisé le package `gbm`, qui utilise le boosting par méthode du gradient sur des algorithmes de régression. Nous avons d'abord cherché à optimiser le nombre d'arbres, les autres paramètres étant fixés, autrement dit, le modèle à estimer et le boosting correspondant sont donnés dans le code suivant :

```
> # Eq <- bikes ~ clouds + humidity + pressure + temperature + wind + month + hour +
> # dow + seasons + factor(is_we) + bikesPrec + time
> # Ntrees <- ...
> #
> # gbm0 <- gbm(Eq, n.trees = Ntrees, interaction.depth = 2, n.minobsinnode = 5,
> #     shrinkage = 0.05, bag.fraction = 0.5, train.fraction = 0.9,
> #     keep.data = FALSE, n.cores = 4, distribution = "gaussian" , data = data0)
> #
> # gbm0$forecast <- predict(gbm0,n.trees=Ntrees,single.tree=FALSE,newdata=data1)
```

Nous obtenons les erreurs de validation croisées en Figure 2.8, montrant qu'à partir de 100 arbres, la performance de l'algorithme de validation croisée ne s'améliore pas significativement.

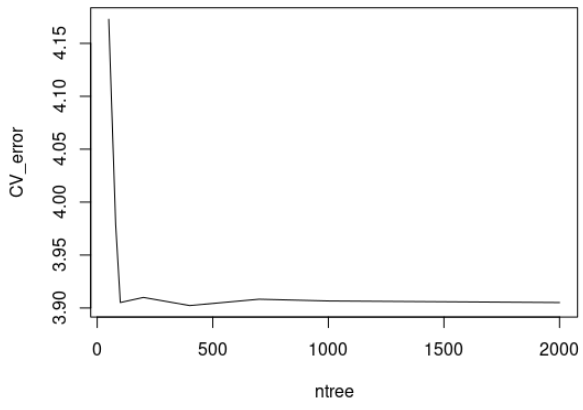

Figure 2.8: Erreur de validation croisée pour l'algorithme gbm en fonction du nombre d'arbres, les autres paramètres étant fixés à des valeurs standard.

Puis nous avons optimisé le paramètre `shrinkage`, en fixant le nombre d'arbres à 100. Nous obtenons les erreurs de validation croisée en Figure 2.9. Le fait que la performance soit insatisfaisante pour de petites valeurs du `shrinkage` nous a étonné puisque nous savons qu'il est en général conseillé de réduire le `shrinkage` autant que possible. Le modèle a également une performance comparable à celle d'autres modèles type additif avec une `rmse` de test de 3.99. Un autre point positif est l'intérêt de ce modèle pour l'agrégation d'experts (voir Section 2.3 et notamment Figure 2.17).

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

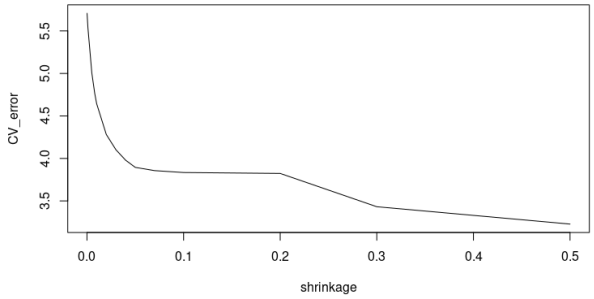

Figure 2.9: Erreur de validation croisée pour l'algorithme gbm en fonction du shrinkage, le nombre d'arbres étant fixé à des 100.

Nous avons aussi tenté de trouver le shrinkage optimal pour 1000 arbres, voir Figure 2.10. Les résultats sont similaires à ceux avec 100 arbres.

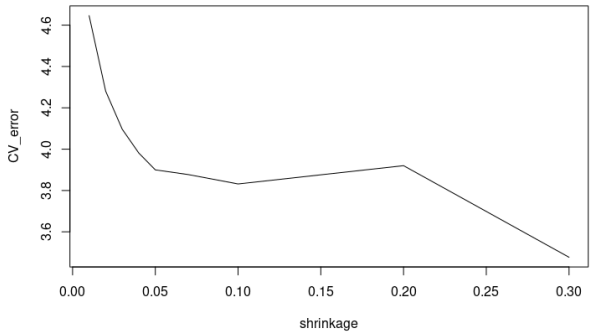

Figure 2.10: Erreur de validation croisée pour l'algorithme gbm en fonction du shrinkage, le nombre d'arbres étant fixé à des 1000.

Afin d'augmenter la performance du modèle en utilisant un modèle ARIMA sur ses résidus, nous nous sommes limités à 100 arbres afin que les temps de calcul restent maîtrisés. La procédure suivie est la même que pour les modèles additifs précédents et la fonction `auto.arima` suggère un modèle ARIMA(4,1,0). La `rmse` de test est de 1.47, soit 9.2%, ce qui est assez satisfaisant (voir Figure 2.11).

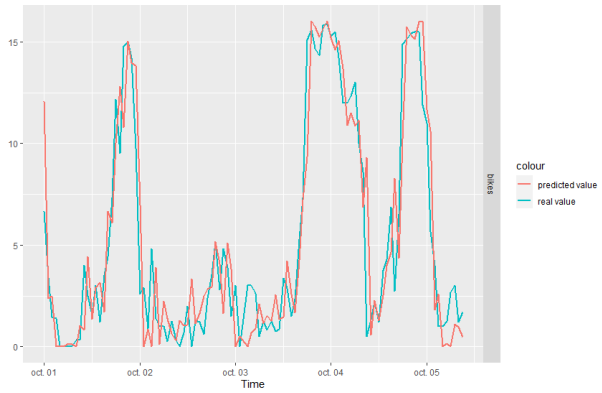

Figure 2.11: Comparaison entre valeur réelle et prédiction par la méthode de gradient boosting sur la première semaine d'Octobre.

2.2.2 Forêts aléatoires

La plupart des travaux sur la prédiction pour les vélos en libre service que nous avons trouvés, utilisent des forêts aléatoires. Nous avons donc utilisé ce modèle (Breiman, 2001), qui moyenne les prédictions de centaines d'arbres de régression, pour pouvoir améliorer notre prédiction. Nous avons travaillé à partir du package `randomForest`. Nous avons d'abord utilisé les forêts sur les données initiales, mais pour les mêmes raisons que dans la partie GAM précédente, nous avons finalement décidé de travailler avec les données agrégées par heure. L'entraînement de forêts aléatoires étant assez long, nous avons cherché un moyen de réduire le temps de calcul de nos fonctions. Le fait d'agréger nos données par heure ce qui a énormément réduit la taille de nos données, diminuant le temps de calcul. Nous avons aussi voulu réduire le nombre d'arbres générés par le modèle, nous avons donc tracé la Figure 2.12. Nous nous sommes donc rendus compte que l'erreur produite par le

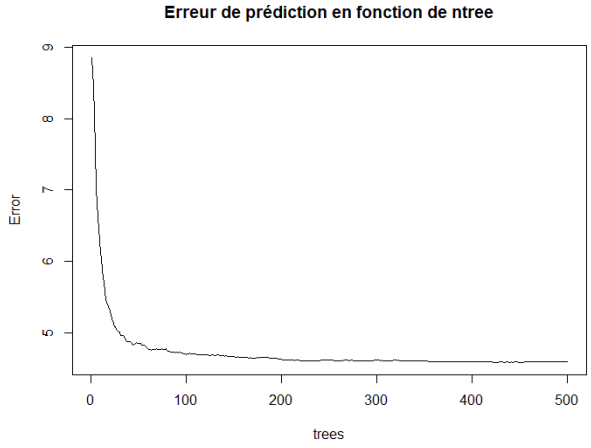

Figure 2.12: Erreur de validation croisée pour le modèle de forêt aléatoire en fonction du nombre d'arbres créées

modèle n'évolue plus au-delà de 200 arbres générés, nous avons limité le nombre d'arbres générés par `randomForest` de 500 à 200. Nous avons donc fixé la valeur `ntree` à 200 et `mtry` $= 4$, le nombre de variables à tester à chaque coupe. Tout ceci nous a permis d'obtenir des procédures s'effectuant en temps raisonnable. En utilisant le modèle naïvement, en considérant toutes les variables, nous obtenons un `rmse` de 2.6. Les forêts aléatoires sont donc bien plus performantes que les modèles additifs, ce qui explique qu'elles soient beaucoup utilisées en pratique. Pourtant, on peut encore améliorer le modèle car il reste encore de l'auto-corrélation dans les résidus. Comme pour les modèles additifs, nous allons

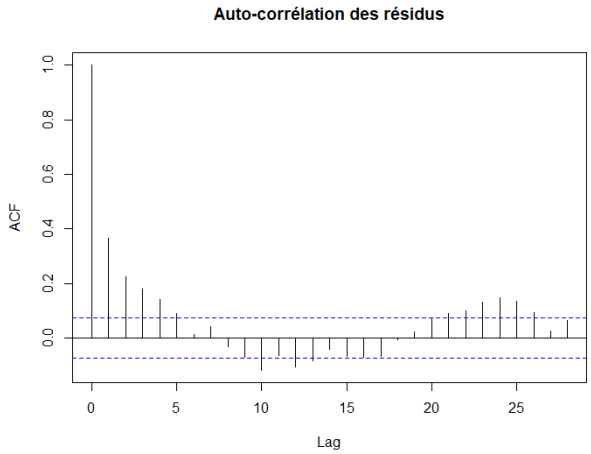

Figure 2.13: Auto-corrélation des résidus de la prédiction le modèle de forêt aléatoire

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

utiliser des séries temporelles pour se débarrasser de cette auto-corrélation, `auto.arima` nous propose d'utiliser un modèle `ARIMA(3,0,1)`.
On obtient finalement un `rmse` de 2.4, le modèle n'a pas été beaucoup amélioré mais on obtient une performance proche de celle de la partie GAM précédente.
On remarque que la prévision (Figure 2.14) reproduit assez fidèlement la tendance de la courbe à approximer mais le modèle à du mal à prédire les valeurs extrêmes, la prévision à l'air toutefois moins précise que celle obtenue par les modèles additifs généralisés.

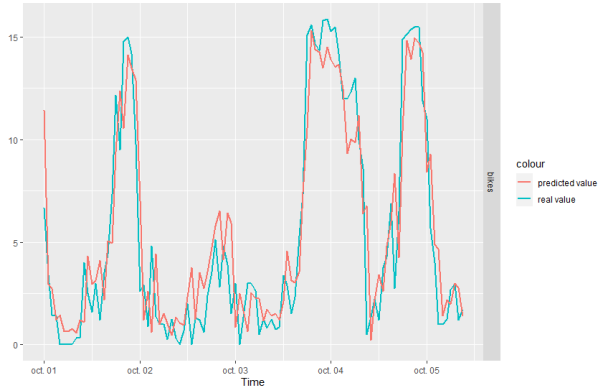

Figure 2.14: Comparaison entre valeur réelle et prédiction par le modèle de forêt aléatoire pour la première semaine d'Octobre

2.3 Agrégation d'experts

2.3.1 Une autre approche séquentielle

Dans le contexte de notre travail, l'agrégation d'experts consiste à se donner un ensemble de modèles entraînés (les "experts"), et de considérer un prédicteur qui s'exprime comme une combinaison convexe des prédictions de ces experts : on cherche à optimiser les poids de cette combinaison convexe, à partir de l'observation des performances des experts au fil du temps (sur les données de test). Comme expliqué en Section 1.1.3, toute notre étude se place dans le cadre d'une approche séquentielle. L'agrégation d'experts s'inscrit dans ce cadre : bien que nos experts individuels ne seront pas ré-entraînés séquentiellement, les poids de la combinaison convexe seront optimisés séquentiellement. Mentionnons que l'écart temporel entre deux recalibrages des poids ne sera pas à l'ordre du jour : elle sera de l'écart temporel (variable) entre deux emprtunt/retour consécutifs pour les données brutes et de 1h dans le cas des données agrégées par heure.

2.3.2 Démarche d'analyse pour le choix de la stratégie d'agrégation

Nous expliquons ici succintement les raisons qui nous ont poussé à choisir notre stratégie finale d'agrégation. Cela nécessite au préalable un rappel de la démarche d'analyse usuelle (dont nous avons une modeste connaissance issue de certains cours). Nous renvoyons vers Stoltz (2010) pour une présentation générale. On appellera dans la suite "poids dégénérés" un vecteur de poids convexes dont une coordonnée vaut 1. On notera $\ell_t(p_t)$ la perte quadratique à l'étape t de l'estimateur issu de l'agrégation des experts selon le vecteur de poids convexes p_t. On peut, a posteriori, définir quelle "aurait été" la stratégie optimale au sein d'une famille de stratégies donnée. Ainsi, pour une stratégie, on mesure naturellement notre degré de satisfaction a posteriori, ou "regret", en comparant les pertes obtenues

aux pertes qui auraient été obtenues si nous avions choisi la stratégie optimale. Plus formellement, lorsque l'on choisit comme objectif de minimiser la perte cumulée

$$L_T(p) := \sum_{t=1}^{T} \ell_t(p_t),$$

le regret se définit comme:

$$R_T(p) := \sum_{t=1}^{T} \ell_t(p_t) - \inf_{\tilde{p}\in\mathcal{F}} \sum_{t=1}^{T} \ell_t(\tilde{p}_t),$$

où $\mathcal{F}$ est une certaine famille de stratégies. Comme les fonctions de perte ℓ_t sont convexes en les p_t (ce sont des formes quadratiques) et bornées (car le nombre de vélos l'est), on peut montrer que lorsque la famille $\mathcal{F}$ est la famille des stratégies constantes (mêmes poids à chaque étape), il existe des stratégies définies sans spécification d'un nombre (fini) T d'étapes, et donnant dans le pire cas un regret au plus de l'ordre de $\sqrt{T}$ lorsque $T \to \infty$. C'est pourquoi un regret linéaire en T est considéré comme une très mauvaise performance.
Venons-en à notre première (petite) décision. Il peut être montré que la stratégie consistant à choisir à chaque étape un vecteur de poids dégénéré sélectionnant l'expert qui a la meilleure perte cumulée, est une stratégie qui peut conduire à un regret linéaire en T dans certains cas. Il semble donc qu'il convient de toujours "laisser sa chance à chaque expert". C'est pourquoi nous avons décidé de ne pas expérimenter cette stratégie.

Dans l'esprit de cette remarque, une idée popualire est de choisir pour chaque expert j, un poids (non nul) proportionnel à une exponentielle d'une quantité spécifique à cet expert, par exemple la stratégie EWA :

$$p_t^{(j)} := C * \exp\left(-\eta L_{t-1}(\delta_j)\right),$$

où δ_j désigne simplement le poids dégénré sélectionnant l'expert j, η est un réel strictement positif et C une constante de normalisation. Dans cette méthode et d'autres méthodes similaires, lorsque η ne dépend pas de temps, ce paramètre est appelé vitesse d'apprentissage.

Compte tenu de la nature de notre problème de prévision, où la saison influe grandement sur le vélo-partage, la question de l'adaptation de notre prédicteur aux changements d'usage était fondamentale. Ceci est vrai dans un contexte applicatif mais est d'autant plus vrai dans le cadre de notre travail où nous n'observons que 5 mois de données, d'avril à septembre, ce qui n'est pas nécessairement représentatif des conditions climatiques du mois d'octobre. C'est pourquoi nous avons d'abord décidé d'utiliser une certaine classe de méthodes où le choix du poids $p_t^{(j)}$ est basé non pas sur les pertes $\ell_s(\delta_j)$ (avec $s \in \{1, ..., t-1\}$) mais sur les dérivées partielles $\partial_j \ell_s(p_s)$. En effet, ceci permet de repérer un expert j dont la performance serait en train de s'améliorer. De plus, nous avions besoin de la vitesse d'apprentissage puisse varier au cours du temps et s'adapter ainsi aux changements d'usage. Ceci nous a conduit à la méthode MLpol. Celui-ci est disponible dans le package `opera` via la fonction `mixture`.
Commençons par les données non agrégées par heure. Nous avons d'abord utilisé tous nos modèles comme experts. C'est alors que nous avons retrouvé le fait que la plupart des modèles augmentés d'un prédicteur ARIMA sur les résidus se distinguaient des autres (voir Figure 2.15). La `rmse` de l'algorithme d'agrégation sur le mois de test est de 1.45 vélos (voir Figure 2.16) sur 16 soit une erreur moyenne de 9%. Ceci est très légèrement mieux que le meilleur expert, le réseau de neurones augmenté d'un prédicteur ARIMA sur ses résidus, dont la `rmse` est de 1.46.

Signalons que nous avions, au début du projet, choisi comme ensemble d'entraînement seulement la dernière semaine, et dans ce cas le classement des experts était très différent et nous obtenions les résultats de la Figure 2.17. Cela nous permet notamment de prendre conscience de la variabilité induite par le fait de travailler sur un jeu de données relativement petit, ce qui est notamment une des difficultés du challenge. Nous clôturons maintenant ce petit aparté.

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

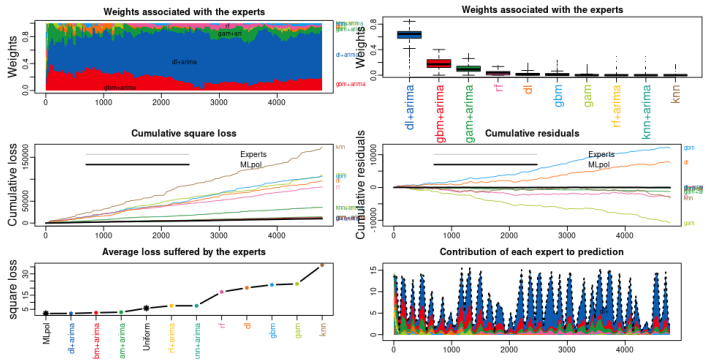

Figure 2.15: Résultat de l'agrégation d'experts sur le mois de test des données non agrégées par heure, à partir de modèles entraînés sur les 5 premiers mois.

```
> summary(agg.online)
Aggregation rule: MLpol
Loss function:  square loss
Gradient trick:  TRUE
Coefficients:
 gam gam+arima    gbm gbm+arima     rf rf+arima dl dl+arima knn knn+arima
   0    0.0805 0.0238     0.186 0.0303        0  0     0.68   0         0

Number of experts:  10
Number of observations:  4779
Dimension of the data:  1

        rmse mape
MLpol   1.45  Inf
Uniform 2.41  Inf
```

Figure 2.16: Résultat de l'agrégation d'experts sur le mois de test des données non agrégées par heure, à partir de modèles entraînés sur les 5 premiers mois.

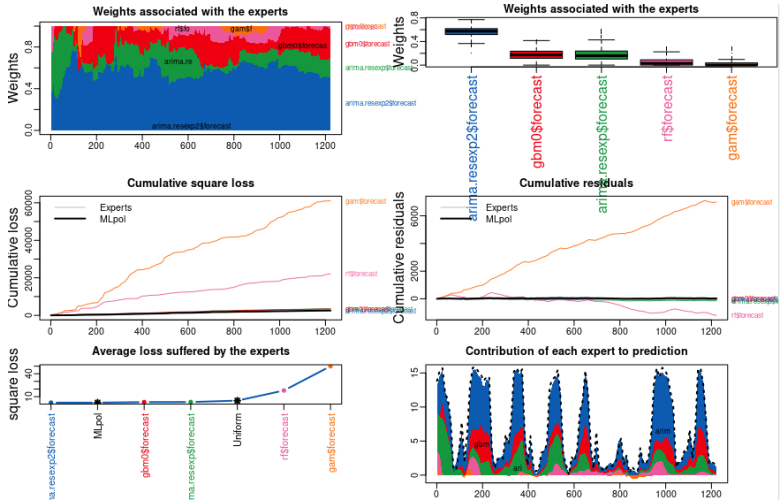

Figure 2.17: Résultat de l'agrégation d'experts sur le mois de test des données non agrégées par heure, à partir de modèles entraînés sur les 5 premiers mois. Les prédicteurs arimaresexp et arimaresexp2 désignent les experts gamArima et rfArima respectivement.

Nous avons ensuite tenté de ne conserver comme experts que les modèles les plus performants (Figure 2.18). Ceci n'améliore pas les performances, voir Figure 2.19.

Avec les données agrégées par heure nous obtenons un classement différent (voir Figure 2.20) et une `rmse` de 2.33 (voir Figure 2.21), le meilleur expert ayant une `rmse` de 2.46. Nous constatons donc que le recours à l'agrégation fait passer l'erreur moyenne de 15.6% à 14.6%.

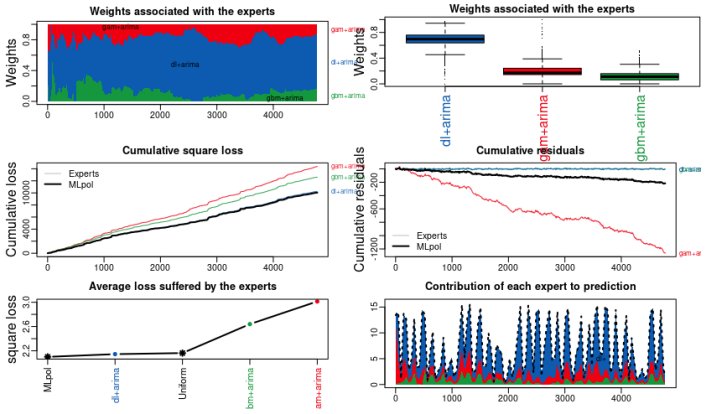

Figure 2.18: Résultat de l'agrégation des meilleurs experts sur le mois de test des données non agrégées par heure, à partir de modèles entraînés sur les 5 premiers mois.

```
> summary(agg.online)
Aggregation rule: MLpol
Loss function:  square loss
Gradient trick:  TRUE
Coefficients:
 gam+arima gbm+arima dl+arima
     0.148     0.164    0.688

Number of experts:  3
Number of observations:  4779
Dimension of the data:  1

        rmse mape
MLpol   1.45  Inf
Uniform 1.47  Inf
```

Figure 2.19: Résultat de l'agrégation des meilleurs experts sur le mois de test des données non agrégées par heure, à partir de modèles entraînés sur les 5 premiers mois.

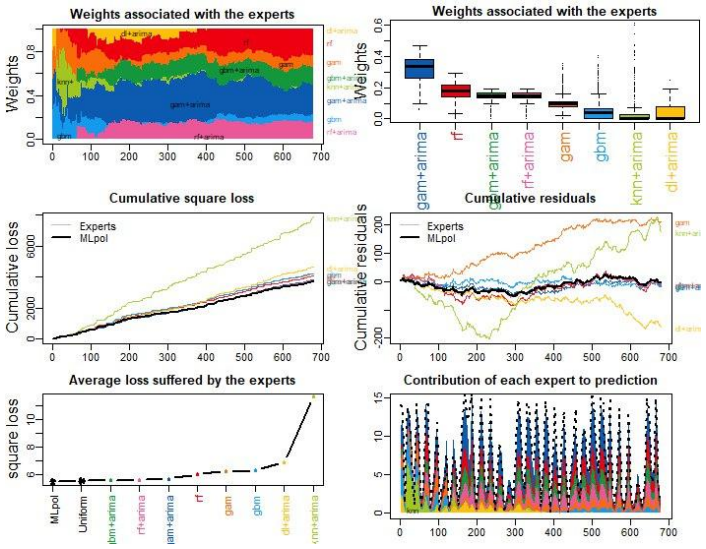

Figure 2.20: Résultat de l'agrégation d'experts sur le mois de test des données agrégées par heure, à partir de modèles entraînés sur les 5 premiers mois.

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

```
> summary(agg.online)
Aggregation rule: MLpol
Loss function:  square loss
Gradient trick:  TRUE
Coefficients:
 gam+arima    rf   gam    gbm gbm+arima rf+arima dl+arima knn+arima
     0.261 0.236 0.114 0.0717     0.159    0.159        0         0

Number of experts:  8
Number of observations:  679
Dimension of the data:  1

        rmse mape
MLpol   2.33  Inf
Uniform 2.34  Inf
```

Figure 2.21: Résultat de l'agrégation d'experts sur le mois de test des données agrégées par heure, à partir de modèles entraînés sur les 5 premiers mois.

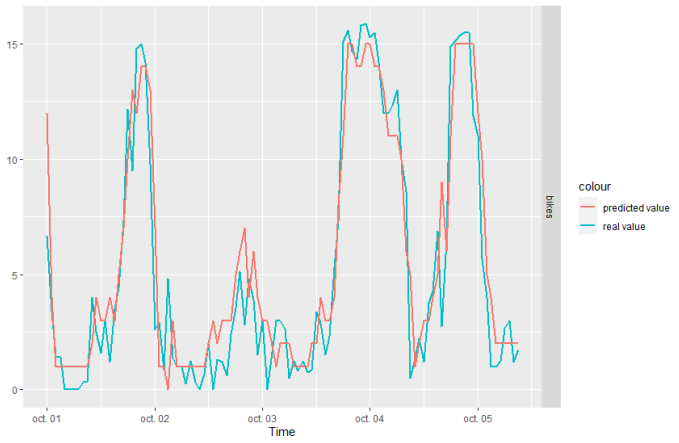

Figure 2.22: Comparaison entre valeur réelle et prédiction par l'agrégation d'experts sur la première semaine d'Octobre.

Modèle	rmse
GAM	1.6
Random Forest	2.4
Gradient Boosting	1.47
Réseaux de neurones	1.46
Agrégation d'experts	1.45

Table 2.1: Rmse des différents modèles sur les données agrégées par heure

Conclusion

En débutant ce projet, nous avions comme ambition la prédiction du flux de vélos dans une station à n'importe quelle moment de la journée. Nous avons vu dans ce rapport que certaines difficultés liées en particulier à la modélisation de la dépendance temporelle nous ont poussées à revoir nos objectifs. Nous avons donc décidé de nous pencher sur la prédiction heure par heure et de manière séquentielle. Cela nous a mené à transformer les données en les agrégeant en heure et à retravailler tous les modèles étudiés sur les données brutes.
Lors de nos travaux, nous nous sommes rendus compte que les performances (en termes de `rmse`, voir la Table 2.1) obtenus pour chacun des différents modèles testés relevaient du même ordre, mais en étudiant plus précisément les résidus, nous avons remarqué que ces modèles n'effectuaient pas les mêmes erreurs de prédiction. Nous également avons testé d'autres modèles non retenus dans ce rapport par soucis de temps de calcul et de moins bonne performance, en particulier les k-plus proches voisins. Dans ce travail, la combinaison de la méthode de gradient boosting et des séries temporelles (ARIMA) paraît légèrement meilleur que les autres modèles, mais souvenons nous que ces résultats ont été obtenus à partir d'une seule station. En considérant plusieurs stations, nous nous sommes rendus compte qu'aucun modèle ne sortait du lot et que l'erreur de prédiction de l'agrégation de modèles donnait une certaine amélioration et variait entre 8% et 15 %. Nous avons donc décidé d'agréger les modèles étudiés et nous avons effectivement réussi à réduire les erreurs de prédiction. Cette agrégation des modèles constitue alors le modèle retenu, car il s'est avéré être le meilleur sur toutes les stations que nous avons testées. Ce modèle reste pourtant perfectible, ses erreurs de prédiction les plus flagrantes se font souvent sur les valeurs extrêmes, il serait donc possible d'améliorer le modèle en utilisant une méthode précise sur les valeurs extrêmes. La prise en compte de la dimension spatiale dans la modélisation permettant de tenir compte de la situation des stations voisines serait une idée de piste à explorer.

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

Bibliography

Borgnat, P., Robardet, C., Abry, P., Flandrin, P., Rouquier, J.-B., and Tremblay, N. (2013). A dynamical network view of lyon's vélo'v shared bicycle system. In *Dynamics On and Of Complex Networks, Volume 2*, pages 267–284. Springer.

Bouveyron, C., Côme, E., Jacques, J., et al. (2015). The discriminative functional mixture model for a comparative analysis of bike sharing systems. *Annals of Applied Statistics*, 9(4):1726–1760.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Etienne, C. and Latifa, O. (2014). Model-based count series clustering for bike sharing system usage mining: a case study with the vélib'system of paris. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):1–21.

Faghih-Imani, A. and Eluru, N. (2016). Incorporating the impact of spatio-temporal interactions on bicycle sharing system demand: A case study of new york citibike system. *Journal of Transport Geography*, 54:218–227.

Fasiolo, M., Wood, S. N., Zaffran, M., Nedellec, R., and Goude, Y. (2020). Fast calibrated additive quantile regression. *Journal of the American Statistical Association*, pages 1–11.

Fishman, E. (2016). Bikeshare: A review of recent literature. *Transport Reviews*, 36(1):92–113.

Froehlich, J. E., Neumann, J., and Oliver, N. (2009). Sensing and predicting the pulse of the city through shared bicycling. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Gervini, D. and Khanal, M. (2019). Exploring patterns of demand in bike sharing systems via replicated point process models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 68(3):585–602.

Han, K., Müller, H.-G., Park, B. U., et al. (2018). Smooth backfitting for additive modeling with small errors-in-variables, with an application to additive functional regression for multiple predictor functions. *Bernoulli*, 24(2):1233–1265.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized additive models*, volume 43. CRC press.

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Li, Z. and Wood, S. N. (2020). Faster model matrix crossproducts for large generalized linear models with discretized covariates. *Statistics and Computing*, 30(1):19–25.

Saberi, M., Ghamami, M., Gu, Y., Shojaei, M. H. S., and Fishman, E. (2018). Understanding the impacts of a public transit disruption on bicycle sharing mobility patterns: A case of tube strike in london. *Journal of Transport Geography*, 66:154–166.

Stoltz, G. (2010). Agrégation séquentielle de prédicteurs : méthodologie générale et applications à la prévision de la qualité de l'air et à celle de la consommation électrique. *Journal de la Société Française de Statistiques*, 151 No. 2:66–106.

Torti, A., Pini, A., and Vantini, S. (2021). Modelling time-varying mobility flows using function-on-function regression: Analysis of a bike sharing system in the city of milan. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 70(1):226–247.

Wood, S. and Wood, M. S. (2015). Package 'mgcv'. *R package version*, 1:29.

Wood, S. N., Goude, Y., and Shaw, S. (2015). Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C: Applied Statistics*, pages 139–155.

Zhou, X. (2015). Understanding spatiotemporal patterns of biking behavior by analyzing massive bike sharing data in chicago. *PloS one*, 10(10):e0137922.

Issa-Mbenard Dabo et Yassine Hamdi ©UPSaclay
issa-mbenard.dabo@ens-rennes.fr,yassine.hamdi@polytechnique.edu

Annexes

Le package R regroupant l'ensemble des codes et algorithmes permettant de réaliser les travaux est disponible à ce lien Package R.

L'interface graphique interactive permettant d'explorer les series de locations de vélos de 30 stations à Lyon, Paris, Toulouse est disponible à ce lien Interface Rshiny.