Building Appliances : Home Saver

Réalisé par
Bertrand PATUREL et Cyril NEFZAOUI

SIM202

Rapport

ENSTA Paris

Rapport donné à Yannig GOUDE
Date de remise : 29/03/2020

Table des matières

Résumé

Ce rapport présente les prévisions de l'équipe Home Saver de la consommation globale des appareils dans une maison à faible consommation d'énergie en Belgique. Les variables explicatives sont les mesures de température et d'humidité provenant des capteurs de la maison, ainsi que les données météorologiques d'une station voisine et la consommation globale du pays.

Il y figure les résultats et l'analyse de 5 modèles statistiques entrainés à partir des données d'entrainement : La regression linéaire, le modèle GAM, XGBoost, GBM et plusieurs forêts aléatoires. Ce rapport n'a donc pas vocation à présenter les codes. La prédiction la plus précise (celle qui obtient le meilleur score au sens du rmse) est une prédiction linéaire avec classification en fonction des variables humidité et température de la cuisine et de la salle de bain.

1 Description des données

On dispose de deux jeux de données :

- Le tableau de données **train** pour s'entrainer à prédire la consommation de la maison (Appliances) en fonctions des variables explicatives (date,températures...). Ce tableau contient 13964 relevés de 43 variables (soit 42 variables explicatives).

- Le tableau de données **test** contenant toutes les variables explicatives, il n'y a pas la variable Appliances qui est à prévoir. Ce tableau contient 5771 relevés de 43 variables.

Les variables explicatives des deux jeux de données comprennent des mesures au sein de la maison comme des relevés de températures et de l'humidité dans chaque pièce mais aussi des variables externes comme la pression ou l'humidité de la station la plus proche. Les deux jeux de données ont aussi les mesures de la consommation électrique du pays. Toutes ces variables sont associées à des relevés prises à plusiseurs lapses de temps (une des variables naturellement introduite est donc le temps).

2 Visualisation des données

On voit clairement que l'on doit compléter premièrement la consommation en énergie des relevés du test manquants à partir des relevés complémentaires sur le train du 11 janvier 2016 au 20 mai 2016. Cette première partie se fera par *imputation statistique*. Puis l'on doit prédire les consommations de l'énergie sur le reste des dates du test, partie qui est mise en valeur ci dessous en rouge sur le graphe. Cette deuxième partie concerne *la prédiction*.

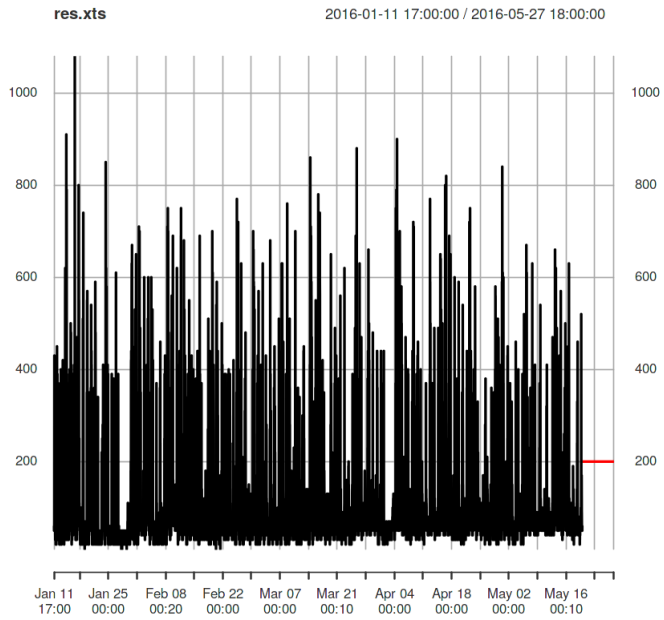

Figure 1 - Mise en évidence de la partie à imputer en noir et de la partie à prédire en rouge.

Il y avait donc initialement un seul jeu de données avec des relevés toutes les 10 minutes, qui a été divisé en deux jeux de données train et test. Dans les donnees du test, qui comporte 5771 elements, il y a donc 4654 premier relevés compris entre les données du train. On a donc un problème mixte et on va traiter séparément dans la suite le problème d'imputation (4654 valeurs) et le problème de prévision (1117 valeurs).

3 Traitement des données

Nous avons d'abord analysé de manière très qualitative les variables explicatives proposées.Certaines de ces variables ne peuvent pas être des variables explicatives.Par exemple, rv1 et rv2 sont des simulations de variables aléatoires et ne sauraient donc être explicatives.On les retire donc du jeu de données.

Ensuite, nous avons avons analysé le type des données.Ces données sont quantitatives (mesures de grandeurs physique :température,humidité relative...), mais aussi qualitatives (WeekStatus).On a remarqué que les variables qualitatives étaient représentées sous la forme de FACTOR. Chaque variable qualitative est liée à son niveau de facon bijective, il est donc facile de convertir ces facteurs en niveaux numériques. Par exemple, pour la variable $DayofWeek$, on a le codage Lundi=1 ; pondérée 2 ; .... Dimanche=7 (technique du $OneHotEncoding$).

Par ailleurs, nous avons remarqué que de nombreuses variables qualitatives étaient redondantes.D'après le principe de parcimonie, nous avons choisi un un sous-ensemble des variables qualitatives qui résument toute l'information apportée par le jeu de variables qualitatives.En appliquant cette procédure, on se ramène en fait à la notion de statistique exhaustive.

On remarque grace au graphique suivant que les NA relevés par les capteurs d'humidité RH 6 correspondent simplement à une saturation des capteurs. Nous remplacons donc les NA par la valeur minimale que peut mesurer le capteurs correspondant à 10. On voit clairement qu'il y a saturation à 10 :

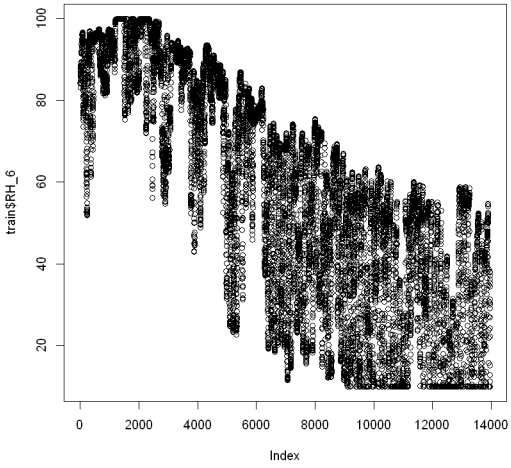

Figure 2 – Les valeurs extremales satures les capteurs de $RH6$.

Pour ce qui de la visibilité, les NA apparaissent de même à cause de la saturation des capteurs, ceux ci ayant du mal à enregistrer des valeurs extrêmes. Ici l'on propose une autre approche pour remplacer les valeurs manquantes. Nous remplacons chaque NA par la moyenne des deux valeurs non NA encadrant notre NA.

4 Imputation statistique par continuité du signal

Puisque nous venons de mettre en évidence qu'il y avait deux parties, une partie comprenant l'imputation statistique et l'autre de la prédiction nous allons nous intérressé dans cette partie l'imputation qui suppose une forme de continuité du signal Appliances en fonction du temps.

Cette imputation se fait de manière suivante :

Notons (A_t) la série de toutes les Appliances, (A_t^e) celle du test à compléter et (A_t^r) celle du train. Pour chaque valeur Appliances (A_t^e) manquante dans le test, on regarde les deux valeurs de train A_{t+k}^r et A_{t-j}^r encadrant et on effectue une moyenne pondérée de ces deux valeurs. Puis on améliore cette aproximation en attribuant un poids différent à ces deux Appliances encadrantes. Ainsi plus la valeur manquante à compléter du test est proche temporellement de son bord gauche (resp droit) plus le poids attribuer au bord gauche est important.

$$\boxed{\mathrm{A}_t^e = \frac{k}{j+k} A_{t-j}^r + \frac{j}{k+j} A_{t+k}^r}$$

Avec j (resp k) le nombre de dizaine de minutes séparant la valeur à compléter A_t^e de son bord gauche A_{t-j}^r (resp de son bord droit A_{t+k}^r).

Remarque : Cette méthode d'imputation est concise et précise (minimise le rmse).

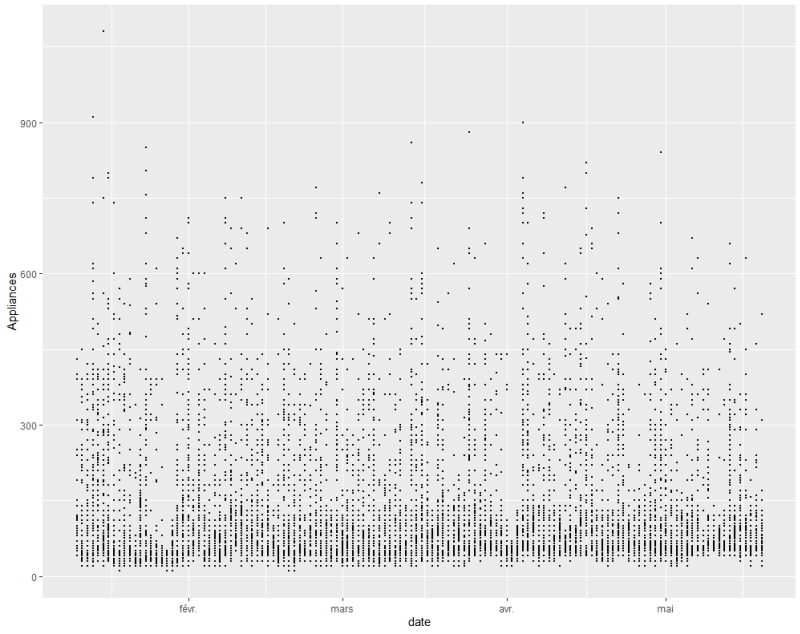

Figure 3 – Les Appliances du test récupérées par imputation en supposant la continuité du signal.

5 Prédiction par regression linéaire

5.1 Choix du meilleur modèle linéaire

On détermine dans un premier temps le degré de liberté du modèle linéaire (id est le nombre de variables adéquates) en caculant la trace de la matrice de projection $X(X^\top X)^{-1}X^\top$. Celle ci vaut 12.

Puis dans un deuxième temps on choisit le modèle linéaire sur les 13000 premières données explicatives et quantitatives en cherchant à minimiser le critère AIC. On utilise donc la fonction StepAIC, on teste les trois modes *forward*, *backward* et *both*. On choisit la recherche *forward* car on controle le nombre de variable plus facilement en choisissant l'option *steps* égale à notre nombre de variables explicatives.

En choisissant 12 variables explicatives l'on a notre rmse de 78.93 entre notre prévision et l'échantillon test corespondant ici aux Appliances de 13000 à 13964 du train. On remarque effectivement par lecture graphique que 12 variables explicatives explicatives minimise bien notre rmse sur ce jeu de données test :

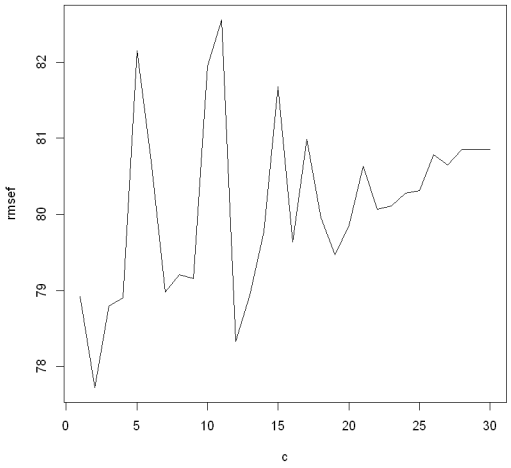

Figure 4 – Evolution du rmse de la prédiction linéaire en fonction du nombre de variables explicatives choisis.

5.2 Prédiction par le modèle linéaire choisit

On retient donc pour notre partie prédiction le modèle linéaire avec 12 degrés de liberté. Pour ce qui est de la partie ou il suffit de copléter les veleurs manquantes du test à partir de celle du train, on choisit de prendre la moyenne pondérée des deux relevés Appliances encadrants les valeurs manquante du test :

En sortie on a en bleu la pratie commune du test et du train ou il suffit juste de compléter

les données manquantes et en vert la prédiction faite par régression linéaire :

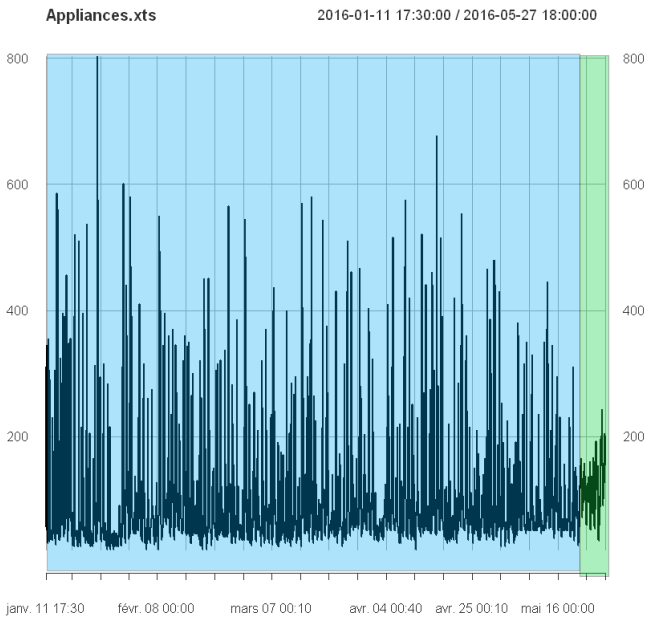

Figure 5 – Mise en exerbe de la partie imputation (en bleu) et de la prédiction linéaire (en vert).

5.3 Amélioration de notre modèle linéaire par classification

Une fois notre meilleur modèle linéaire choisit on décide de l'améliorer en *classifiant* les données. Nous montrons plus tard l'importance de certaines données, notamment la température de la cuisine et son humidité (qui indiquent clairement que la famille est en train de préparer le repas par exemple) ou le température et l'humidité de la salle de bain. Si ces varaiables ont des valeurs fortes nous sommes sur des pics pour appliances et ainsi nous boostons les pics de notre régression linéaire à l'aide d'une méthode proche de celle des **K plus proche voisins**. Ainsi lorsque notre modèle se trouve sur des valeurs extremes pour l'humidité et la température des deux pièces nous lui demandons de remplacer la prédcition linéaire par une moyenne des appliances des pics du à ces deux pièces dans le train.

Nous avons continué cette *classification* en limitant les creux de la prédiction linéaire uniquement sur les périodes où la famille dort. Enfin nous avons codés une fonction qui teste sur différents critères si la famille est partie en dehors de la maison en weekend. Si c'est le cas les valeurs d'appliances sont diminuées.

Sur ce graphique on observe en pointillé les prévisions faite par la prédiction linéaire avant traitement et en ligne les prédictions faites après traitement.

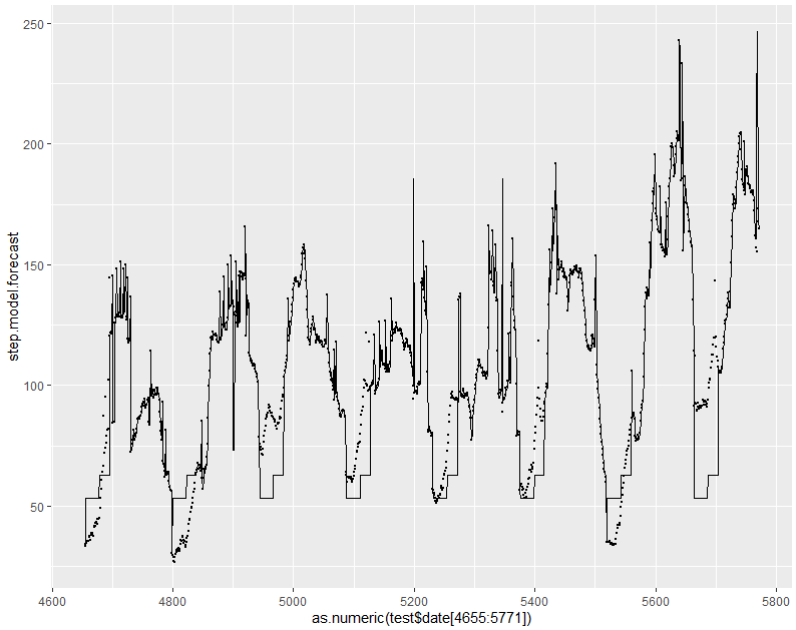

Figure 6 – Comparaison entre le modèle linéaire simple et le modèle linéaire amélioré.

6 Prévision par modèle GAM

6.1 Choix des variables explicatives

Pour choisir nos variables explicatives du modèle de prédiction on peut tout d'abord regarder la corrélation entre Appliances et les variables explicatives adéquates :

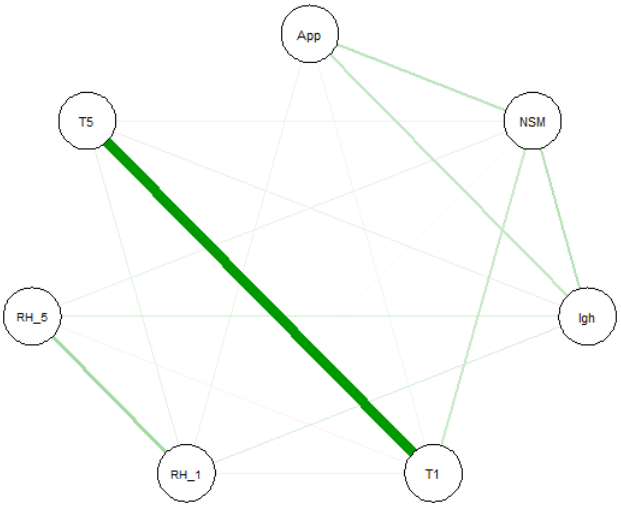

Figure 7 – Corrélation entre certaines variables explicatives et Appliances.

On décide alors de choisir en partie NSM et lights comme variables explicatives (bien que deux variables corrélés n'implique que l'une explique l'autre) pour commencer. Nous avons ensuite complété notre choix de variables avec un StepGam. Voici en noir la courbe reélle d'une partie des valeurs de test ayant servi d'entrainement et en rouge la prédiction de ses valeurs par le gam.

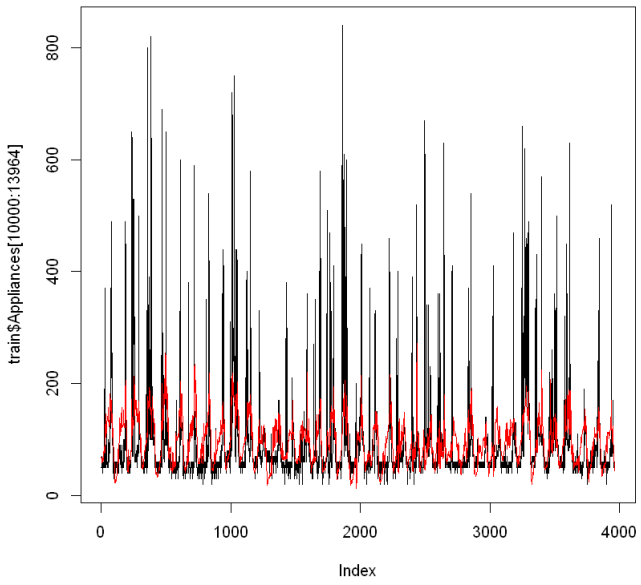

Figure 8 – Comparaison en noir des Appliances servant de test et en rouge de la prédiction de ces Appliances par GAM.

6.2 Modèle mixte : GAM et linéaire

Nous avons améliorer nos deux modèles de prédiction GAM et linéaire en faisant une combinaison de ces mmodèles. L'idée est de laisser le modèle GAM s'exprimer lorsqu'il y a des pics importants et lorsque les habitants dorment (pour combler les défaults).

On effectue donc tout d'abord une prédiction (g_t) avec un modèle gam expliquant les Appliances uniquement par NSM. On souhaite avoir un modèle prédictif valant zéro lorsque les habitants dorment et valant 1 lorsqu'il y a un pic de consommation. On pose donc $G_t = \frac{g_t - min(g_t)}{max(g_t) - min(g_t)}$.

Nos Appliances prédites seront alors (A_t) telque : $\boxed{\mathrm{A}_t = 51 * (1 - G_t) + L_t * G_t}$

où L_t correspond à notre meilleur prédiction linéaire des Appliances manquantes du test.

Justification de ce modèle : Lorsque les habitants dorment, G_t est proche de 0 donc A_t équivaut à 51 ce qui est en moyenne la valeur des Appliances de minuit à 7h00 du matin. Quand il y a un pic G_t est proche de 1 et donc A_t correspon à la prédiction linéaire L_t.

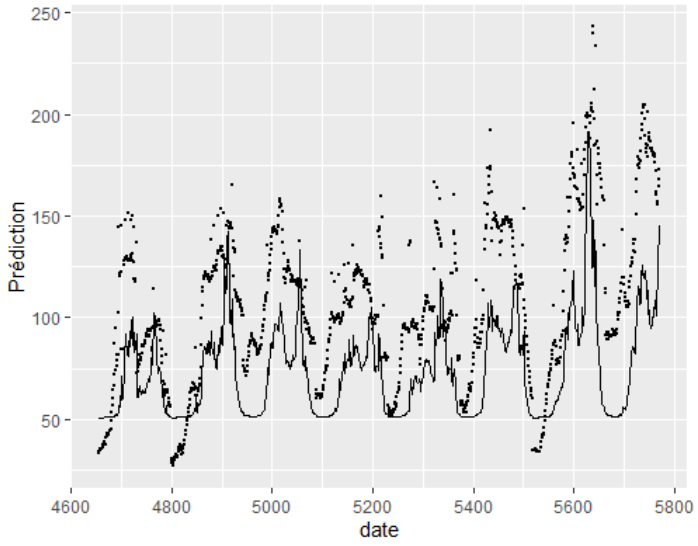

Figure 9 – Comparaison de la régression linéaire (en pointillé) et du mix GAM linéaire.

7 Utilisation d'un GBM (Bootstrap dans le cadre du modèle linéaire)

On utilise une methode de Gradient Boosting Machine (GBM).Nous allons entrainer le GBM sur un modèle linéaire. L'un des avantages de cette méthode est qu'elle repose sur des bases théoriques solides, puisqu'elle s'appuie sur un modèle linéaire et la théorie du réechantillonage avec remise bootstrap. On peut visualiser un indicateurs de la performance : l'erreur Out Of Bag qui doit normalement converger vers 0 quand on augmente le nombre d'itération dans l'algorithme GBM.

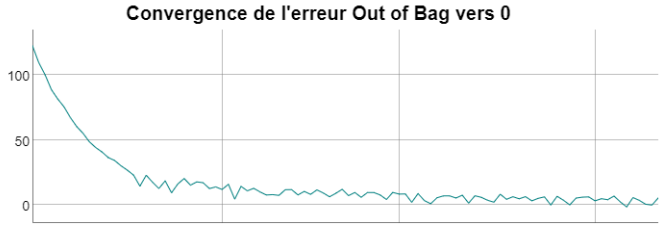

Figure 10 – Convergence de l'OOB vers 0

Definition : L'Erreur Of Bag est $\mathbb{E}((m(X) - Y)^2)$ où $m(X)$ est la prédiction en fonction des variables explicatives X et Y la variable cible (ici Appliances). L'Erreur OOB représente l'écart moyen entre la prédiction et la variable cible, en régression on choisit naturellement pour formule $\frac{1}{n}\sum_{k=1}^{n}(\hat{Y}_k - Y_k)^2$ avec $\hat{Y} = m(X)$.

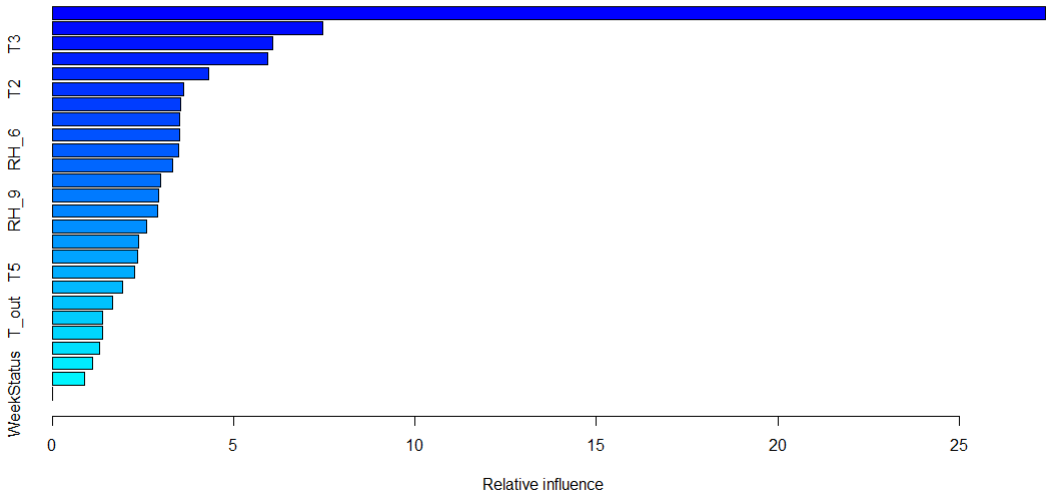

Figure 11 – Représentation de l'influence relative des différentes variables

On remarque ensuite que certaines variables ont une influence relative faible. En appliquant le principe de parcimonie, nous avons donc décidé de retirer ces variables pour ne garder que les plus significatives.

Nous remarquons que ce sont les mêmes variables qui sont retenues dans le cadre de GBM et dans le cadre du modèle linéaire obtenu par *stepAIC*. On peut donc entrainer un modèle linéaire obtenu par step.AIC à l'aide d'un GBM.

8 Utilisation de l'interface XGBoost (Xtreme Gradient Boosting)

Nous avons utilisé XGBoost en tant que GBM amélioré.En effet,XGBoost tout comme GBM utilisent le principe de gradient Boosting, mais il nous a semblé que XGBoost proposait une implémentation plus performante que GBM, grâce à notamment :

- parallélisation du calcul

- tous les paramètres clés de l'apprentissages sont modifiables

- utilisation de matrices pleines pour accélerer le temps de calcul.

- La modélisation a été concue de telle sorte à limiter l'overfitting

Il est possible d'utiliser XGBoost avec un modèle linéaire, mais alors cela revient à utili-

ser un GBM. Notons bien que Xgboost doit permettre de prédire la variable Appliances, qui est quantitative.On utilise donc des arbres de régression, à l'opposition des arbres de décision, classification, segmentation utilisés dans le cadre de la prédiction d'une variable qualitative.

En fait, nous avons utilisé XGBoost comme une interface, qui regroupe en fait plusieurs modèles. Premièrement, on définit une structure de contrôle qui permet de choisir la méthode de validation : nous avons successivement choisi la *validation croisée* (cv) et l'*Erreur Out of Bag* (OOB).

Ensuite, on utilise la fonction expand.grid, qui permet de créer un data frame qui représente toutes les combinaisons possibles des facteurs fournis en entrée.

Nous nous sommes alors demandé comment les arbres de régression développés dans la phase d'apprentissage allaient mesurer l'homogénéité dans les données de Appliances, variable quantitative. On considère la variance inter-groupe pour mesurer l'homogénéité : $V_{inter} = \frac{1}{n}(n_1(\overline{y_1} - \overline{y})^2 + n_2(\overline{y_2} - \overline{y})^2)$ avec n_1 le nombre d'individus dans le groupe 1, n_2 le nombre d'individus dans le groupe 2, $y = Appliances$ et $\overline{y_i}$: moyenne empirique de y pour le groupe i.

Nous avons ensuite adapté les paramètres d'apprentissage afin de fitter au mieux le modèle trouvé à nos données. On a choisi :

- nombre d'itérations à effectuer pour le boosting : c(100,1000,10000)

- profondeur d'arbre maximale : c(1,20). Les meilleurs modèles trouvés présentaient systématiquement une profondeur d'arbre de 15.

- subsample : fraction de l'ensemble d'entrainement utilisée pour developper les arbres à chaque itération. On choisit subsample=0.5 pour limiter l'overfitting.

- eta : taux d'apprentissage.Paramètre de contrôle de la vitesse de convergence lors de la descente de gradient pour la fonction de perte. On choisit eta très faible pour limiter l'overfitting.

9 Techniques de prévision hybride, imputation des valeurs manquantes vs prevision pure

Avant tout traitement de train et test, nous avons rajouté ces data frame d'une colonne $isintest$, dont les coefficients valent 1 ou 0 selon que l'élèment associé appartient au train ou au test. On a ensuite fusionné train et test dans une large $dataframe$ que l'on appelle total. On effectue un tri de total par ordre chronologique croissant, ce qui nous permet de visualiser la répartion de certaines valeurs du test dans le train.

On remarque que le test a été echantilloné de manière linéaire dans le train. On dispose donc d'un échantillonage de bonne qualité, car les modèles peuvent être entrainés sur le train, ce qui aurait été plus difficile si les valeurs manquantes de Appliances avaient été réparties sans homogénéité.

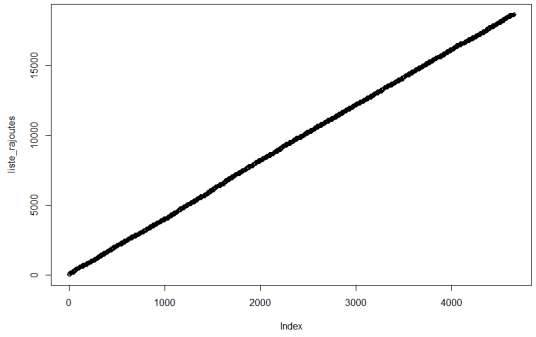

Figure 12 - Mise en évidence de la répartition linéaire des valeurs manquantes dans le jeu d'apprentissage.

Voici le résultat de l'imputation des données manquantes à l'aide d'une méthode de prévision à l'aide de Xgboost en utilisant une méthode de validation croisée.

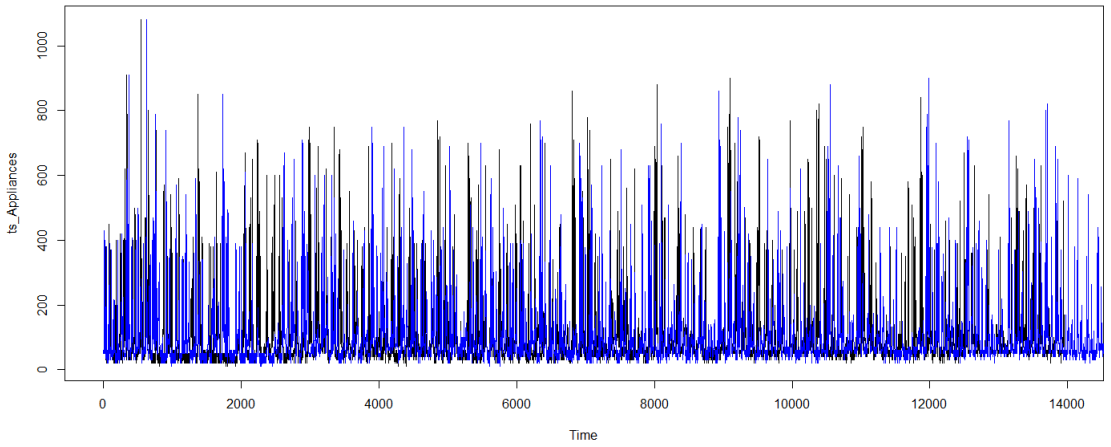

Figure 13 - Imputation des valeurs manquantes dans le jeu d'entrainement via prévision du modèle XGBoost

Par ailleurs, voici le résultat de l'imputation des données manquantes à l'aide d'une méthode de complétion par forêts aléatoires. Plus précisement, nous avons utilisé le *packagemissForest*. Dans cette méthode d'imputation, l'erreur Out of Bag converge vers 0, ce qui indique le validité de notre imputation.

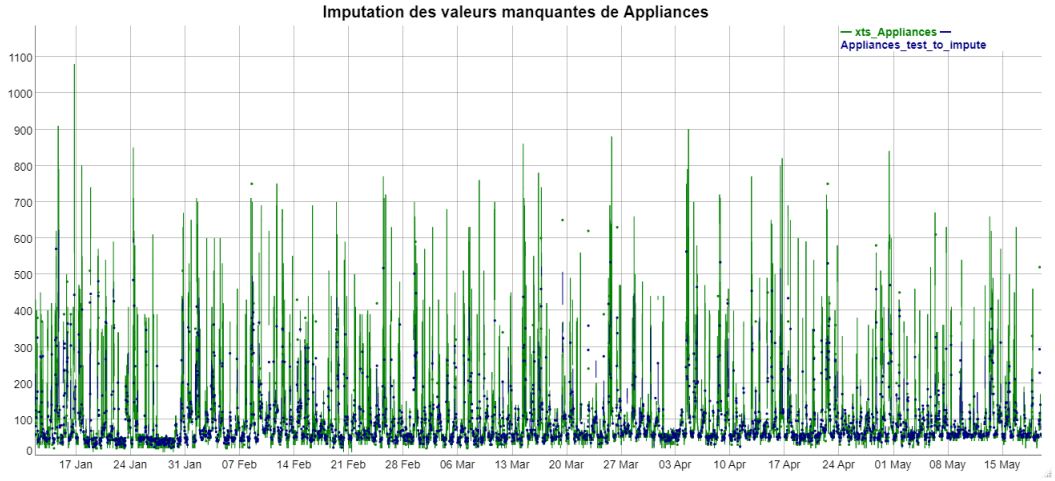

Figure 14 - Imputation des valeurs manquantes dans le jeu d'entrainement par fôrets aléatoires

Nous choisissons de travailler sur les jeu de données entier : on considère que les données imputées font partie des données initiales, donc on peut les réutiliser pour entrainer nos modèles. Nous allons discuter de ce choix dans la suite de notre étude.

9.1 Peut-on réutiliser les valeurs imputées pour améliorer la qualité de notre jeu d'apprentissage ?

Au cours de l'entrainement, nous avons été confronté à un dilemne :Faut-il considérer comme Appliances du train les Appliances qui ont été imputées ? En imputant ces donnees dans le train, on obtient un train complet. On a donc un jeu d'information complet pour la prediction, ce qui est un avantage. Cependant, les Appliances imputées sont des estimées elle-mêmes predites par l'algoritme d'imputation. Elles comportent donc une erreur qui pourrait avoir une repercussion sur la prévision.

A celà s'ajoute une problématique d'échantillonage : Si le jeu de test est bien échantillonné, a priori on peut se passer de ces donnees imputées. Cela aide aussi à ce prémunir de l'overfitting, car le modèle va s'entrainer sur un jeu de données incomplet, donc sera a priori plus apte à prédire le test, qui est aussi un modèle incomplet.

9.2 Tests de nos modèles à l'aide des données imputées

Précedemment, nous avons établi que les données imputées étaient très proches des vraies données au sens du RMSE.Dès lors, on fait l'hypothèse que ces données sont plausibles.On peut donc considérer ces données manquantes avec deux points de vue :

1) Ces données manquantes peuvent être complétées par imputation (ACP par missMDA, forêts aléatoires par missForest)

2)Ces données peuvent être prédites à l'aide d'un modèle qui a été entrainé sans prendre en compte ces données.

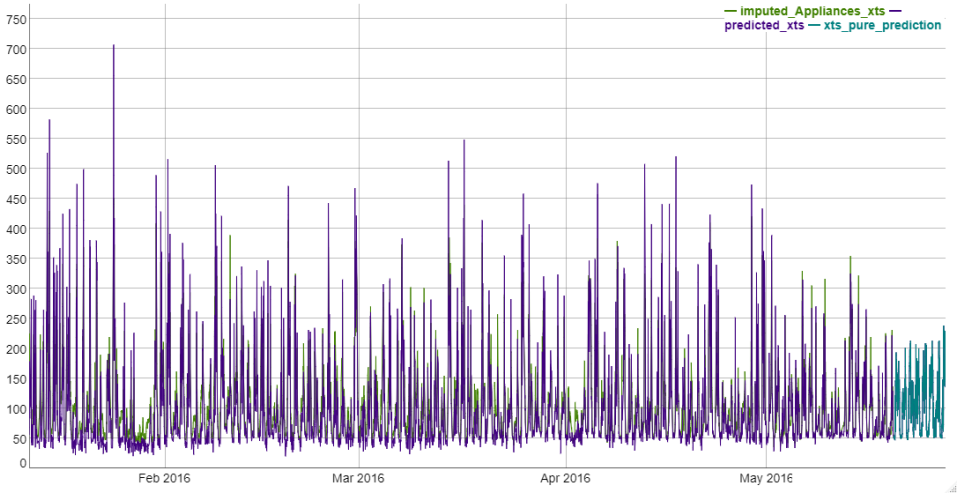

Figure 15 - Imputation puis prévision à l'aide d'un modèle XGboost

Sur cette figure, on a représenté 3 séries mises en jeu dans le test

-en vert, les valeurs imputées par missForest,

-en violet, la partie présente dans le train,

-en bleu,la partie de prédiction pure.

On observe bien une différence de comportement entre les valeurs prédites chronologiquement présentes dans le train et les valeurs correspondant à la prédiction pure. On remarque que le RMSE pour la problème d'imputation est 20, pourtant on obtient un RMSE total de 76 sur Kaggle ;c'est donc que RMSE=56 pour la partie prédiction. On peut donc conclure que pour améliorer notre score, il faut améliorer l'aspect prédiction pure dans notre algorithme.

Nous représentons ici plusieurs prédictions, en utilisant plusieurs jeux de paramètres dans l'algorithme XGBoost

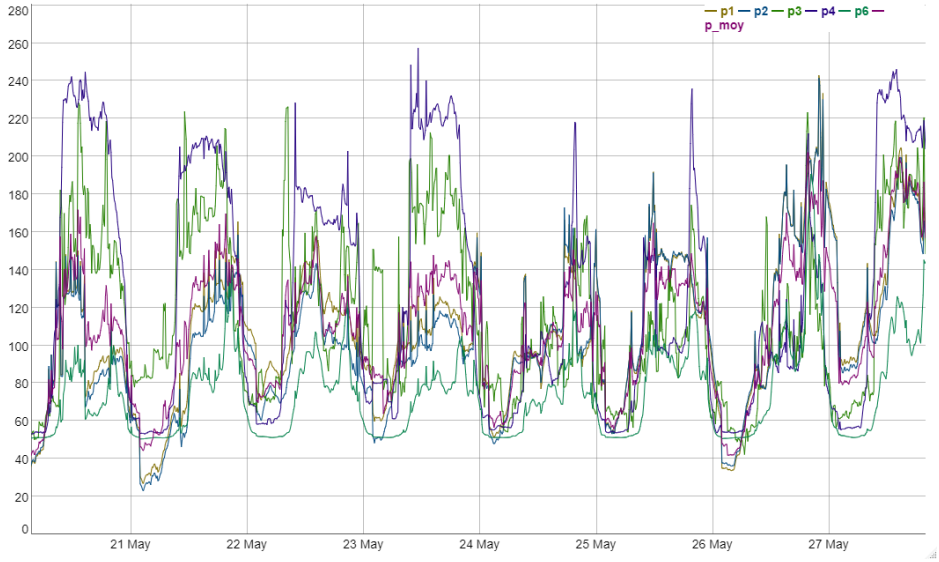

Figure 16 - Superposition de prévisions obtenues à l'aide de plusieurs modèles XGBoost

Il est interessant de remarquer que tous ces modeles ,issus de l'interface xgboost, avec plusieurs jeux de parametres, ont tous la meme allure. Ils présentent tous une allure sous forme de massifs, et non de pics, ce qui est du à un moyennage.Puisque la variable Appliances est numérique, étant donnée une observation x , c'est à dire une réalisation des variables explicatives,qui correspond à une branche dans l'arbre, la prévision de Appliances à partir de x est la moyenne des observations de la branche de x.

De plus, puisque tous les prédicteurs sont numériques,on se ramène en fait à une régression constante par morceaux sur des pavés de R^p,ces pavés étant déterminés par dichotomie succesives lors de la création de l'arbre de régression

Cependant, lors de l'utilisation de XgBoost,nous avons été confronté au problème du surapprentissage.Nous avons choisi nos paramètres pour limiter le surapprentissage :profondeur maximale des branches faibles(20),choix d'un learning rate eta=0.01

De plus, la complexité de l'arbre developpé par xgboost croit de manière exponentielle. Il faut donc limiter la profondeur des branches.Nous avons choisi de limiter la profondeur principale des branches à 15.En effet , le meilleur modèle trouvé ($xgb.train - bestTune$) présentait systèmatiquement une profondeur de 15.la profondeur des branches pouvait varier de 1 à 20 avec un pas de 1.

10 Méthode de permutation-imputation

Nous avons constaté que, une fois le problème d'imputation résolu,le problème de prévision des valeurs dans le futur présente une difficulté supplémentaire :on doit prédire un un ensemble continu dans le temps de Appliances ,alors que dans le problème d'imputation, on pouvait compléter les valeurs manquantes grâce aux valeurs des voisins des Appliances manquantes, par exemple par la méthode des plus proches voisins. Nous avons donc appliqué une permutation

sur l'ensemble d'apprentissage afin d'homogénéisr la répartition des valeurs à prédire de facon uniforme.Ainsi, on transforme un problème de prédiction en un problème d'imputation.

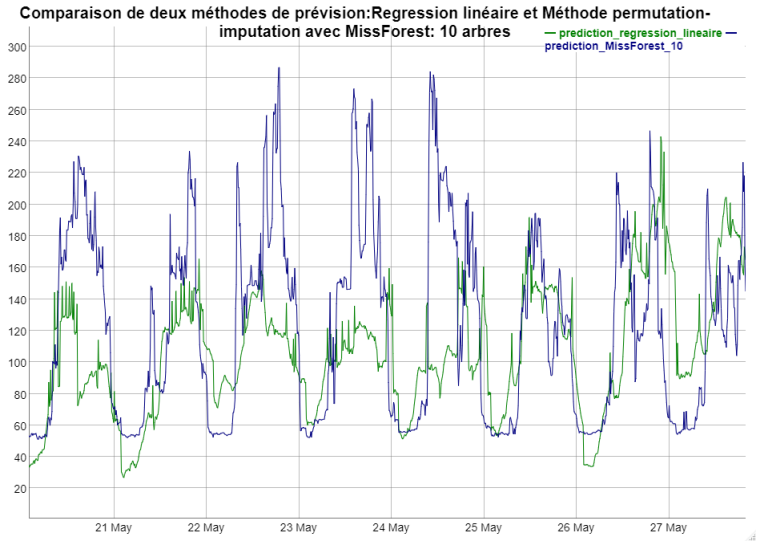

Figure 17 - Superposition de prévision obtenues par régression linéaire / par méthode de permutation-imputation

En vert est représentée une prédiction réalisée à l'aide d'une régression linéaire classique. En bleu est représentée une prédiction réalisée à l'aide de notre méthode de permutation-imputation. Ces deux prévisions obtiennent significativement le même score sur Kaggle, pourtant leur comportement est très différent.La prédiction obtenue par permutation-imputation permet de ne pas sous-estimer les pics grâce à l'utilisation d'un faible nombre d'arbres (10).

Désormais, on a trois possibilités de prédire les valeurs manquantes :

1)Prévision à l'aide d'un modèle (régression, GBM, Xgboost, Random Forest)

2)Imputation, sans permutation.

3)Imputation, avec permutation.

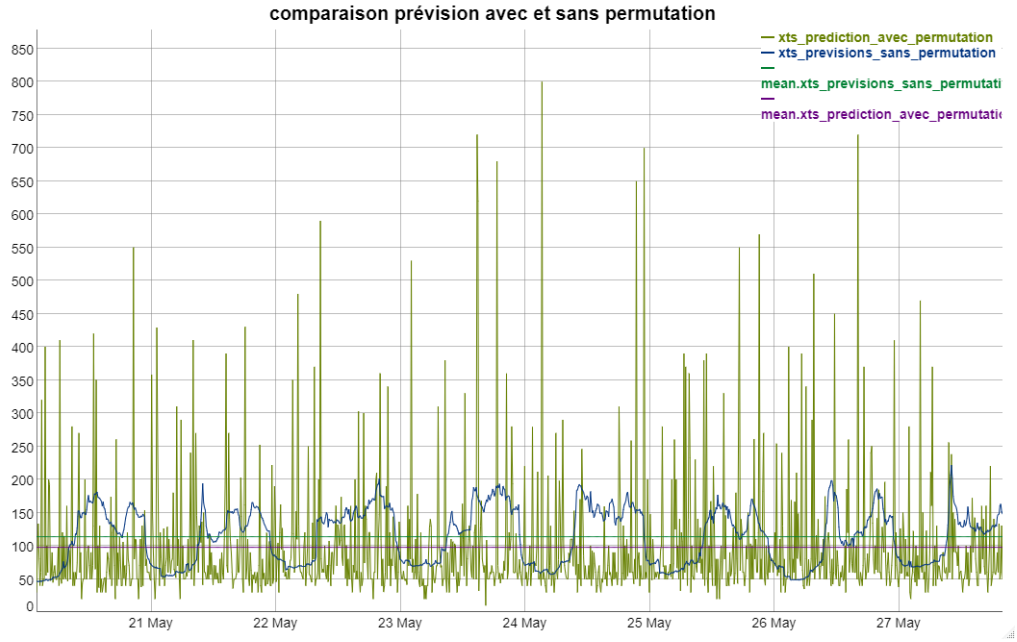

Figure 18 - Mise en évidence de l'influence de la permutation par foret aléatoire.

Interprétation : Sur ce graphe, on superpose les résultats issus d'une même méthode d'imputation, avec et sans permutation. Avec permutation, les valeurs à prédire sont dispersées au sein du jeu de données et le lien temporel entre chaque relevés successif à predire est moindre. Ainsi la variance est plus élevée.

Le RMSE entre ces ceux séries est de 105. La méthode sans permutation fournit un score assez bon, tandis que la méthode avec imputation fournit un score de 84.5, ce qui est moins bon comparé à nos autres performances. En fait, la série temporelle présente une dépendance temporelle trop forte et on doit considérer un ensemble chronologiquement ordonné,sous peine de perdre de l'information.

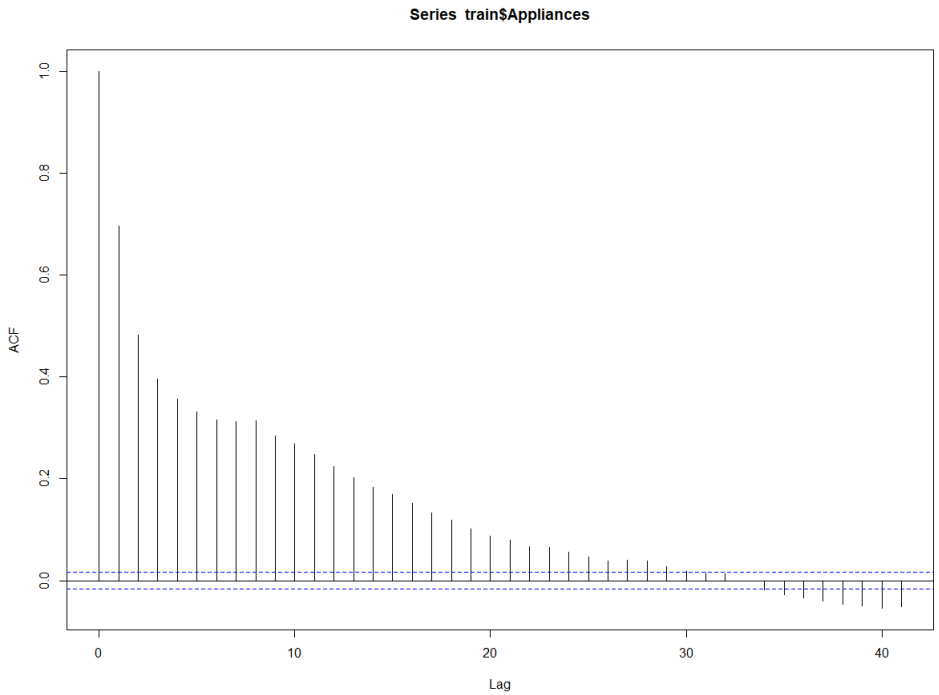

Figure 19 -ACF de la série des Appliances du jeu d'entrainement.

L'ACF de la série des Appliances d'entrainement montre bien une corrélation de la série à son passé.

11 Méthode d'amélioration des résultats :Complétion partielle

On dispose désormais de prévisions de bonne qualité, c'est-à-dire avec un faible RMSE. On choisit nos 3 meilleures prédictions au sens du RMSE, puis on crée la série qui correspond à la moyenne de ces prévisions. Le signal résultant est intéressant car on retrouve de faible appliances la nuit. Ainsi les valeurs d'Appliances très faibles (seuil Appliances <80) paraissent vraisemblables. Nous choisissons donc d'inclure ces individus au sein de notre jeu d'entrainement. On réduit ainsi le cardinal de l'ensemble des données à prédire.En fixant ce seuil , on ne prédit par exemple que 893 valeurs au lieu de 1117. En rouge sont représentés les valeurs de Appliances qu'on introduit dans le jeu d'entrainement, et qu'on a donc plus besoin de prédire.

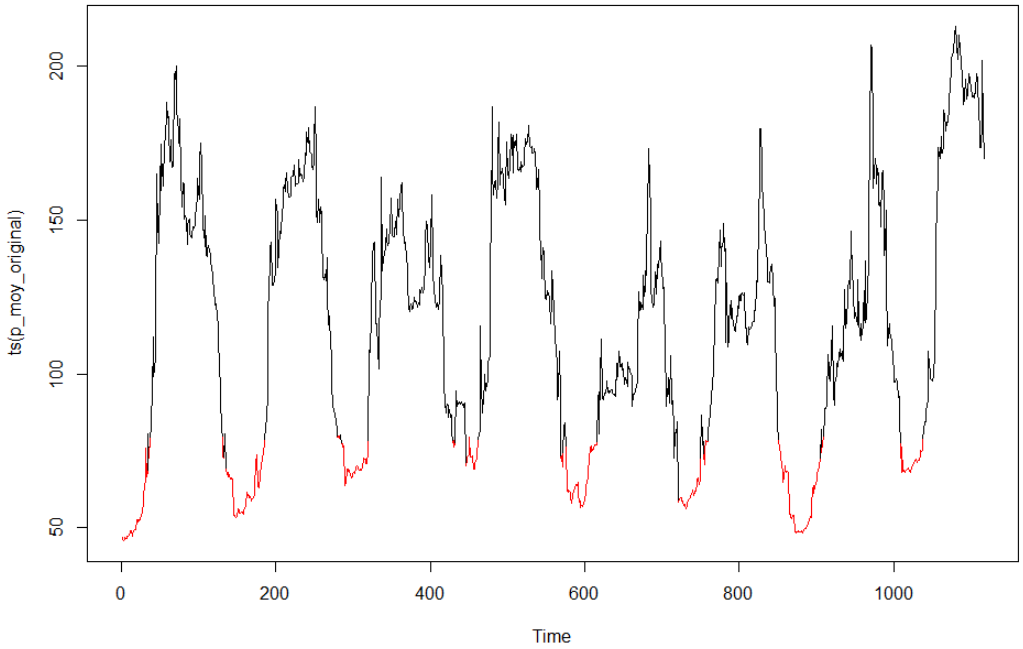

Figure 20 - Complétion partielle en réutilisant des Appliances déjà prédites inférieur à 80

12 Utilisation de Series temporelles

Pour utiliser la Methode de Box-Jenkins dans ce cadre, on a réalisé l'algorithme suivant :

1)Transformer Appliances du train par méthode de Box-Cox.On obtient formellement la série transformée Box-Cox(Appliances).

2)Entrainer le modèle avec Box-Cox(Appliances)

3)Predire.On obtient donc Box-Cox(Prediction)

4)Acceder à la prediction reelle :Inv-Box-Cox(Box-Cox(Prediction))=Prediction

Nous avons initialement pensé que les pics présents dans Appliances étaient un problème pour l'apprentissage. Nous avons donc déduit qu' utilisant des Appliances d'entrainement au variations plus régulières,il serait plus facile d'entrainer le modèle.Pour cette raison, nous avons appliqué la transformation de Box-Cox. Les resultats ne sont pas concluants. En effet, on a obtenu le meme RMSE avec ou sans transformation de Box-Cox. Cette méthode nous a permis de valider qu'il n'y avait pas de problème lors de l'apprentissage, qui serait du à des valeurs très irrégulières et parfois très élevées de Appliances.

13 Tableau récapitulatif de nos résultats

Modèle utilisé	RMSE test (score Kaggle 70 pourcents)
Forêt aléaoire	69.1
GAM	71.8
Mix GAM et regression linéaire	69.5
Regression linéaire avec classification	67.9
Régression linéaire avec imputation par continuité	67.9
Xgboost sans imputation préalable	71
Xgboost avec imputation préalable	69.1
GBM avec modèle linéaire fourni par stepAIC	75.6
transformation Box-Cox+Xgboost	79.4
arbre rpart	100

14 Conclusion

On peut comparer les avantages et les inconvenients de nos diverses méthodes. Dans un premier temps, nous avons mis en place une simple regression linéaire. Cette méthode a le mérite d'être facilement interprétable, et elle constitue une méthode qui conserve une continuité. En effet, de petites variations sur les données conduiront à de petites variations sur les résultats, grâce à la continuité du produit matriciel. Cependant, les régressions linéaires sont par nature inadaptées pour représenter des variations brusques (à savoir les pics).

Au contraire, les méthodes d'arbre permettent de modéliser de facon plus fidèle les discontinuités.

Remarque générale : Lors de l'utilisation de nos modèles, nous avons été confrontés au compromis biais-variance. Des modèles simples, avec peu de variables explicatives ont eu généralement un fort biais mais une variance peu élevée, et les modèles plus complets ont parfois réussi à obtenir un biais inférieur mais une variance élevée.