

SIM 202

Roxane GOFFINET

Mars 2022

**Prédiction de la consommation électrique de l'électroménager en
Belgique**

Table des matières

1	Introduction	2
2	Analyse descriptive	2
2.1	Etat général	3
2.2	Corrélations entre variables	4
2.3	Les caractéristiques des valeurs extrêmes	5
3	Prévisions	7
3.1	Objectifs et critères de validation	7
3.1.1	Erreurs	7
3.2	Première approche par régression linéaire	9
3.3	Forêts aléatoires	10
3.3.1	Principe	10
3.3.2	Premiers modèles	11
3.3.3	Optimisation des paramètres	12
3.3.4	Combinaisons de variables grâce à GAM et CART pour améliorer les performances des forêts	13
4	Conclusion	16

1 Introduction

L'électricité représente en moyenne un quart de l'énergie nécessaire dans un pays. De nombreuses industries reposent sur son utilisation pour fonctionner, elle est un moteur du développement économique. Mais l'électricité sert aussi quotidiennement aux habitants que ce soit pour cuisiner, se chauffer ou encore éclairer son domicile. Il est donc capital de pouvoir répondre quotidiennement aux besoins en électricité. Malheureusement, l'électricité est difficilement stockable, c'est-à-dire qu'il faut produire seulement ce qui sera consommé. C'est pour cela qu'il est très utile de pouvoir effectuer des prédictions sur la consommation électrique à plus ou moins long terme. Nous allons dans ce rapport nous intéresser plus particulièrement à la consommation électrique de l'électroménager d'une famille belge. Dans un premier temps, nous ferons une analyse descriptive des données à notre disposition¹ afin d'en comprendre leur intérêt, nous observerons les corrélations existantes entre certaines variables, et les caractéristiques des valeurs extrêmes que l'on peut observer. Cette partie nous sera utile pour les prédictions que nous effectuerons dans une second temps. Nous utiliserons pour cela plusieurs techniques : régression linéaire, régression ridge, forêts aléatoires etc. Notre but est de trouver un modèle le plus performant possible que l'on pourra comparer à ceux de nos camarades via une compétition kaggle. Finalement, nous conclurons sur les modèles testés et leurs améliorations possibles.

2 Analyse descriptive

Comme mentionné ci-dessus, nous allons dans ce rapport chercher à prévoir la consommation de l'électroménager en Belgique. Nous disposons pour cela d'un jeu de données d'entraînement allant du 11 janvier 2016 au 19 mai 2016. Les données sont enregistrées à intervalles de plus ou moins 10 minutes. Pour chaque enregistrement, nous bénéficions de la date , de la valeur de *Appliances* notre variable à prédire, et des variables explicatives : *lights*, *T1*, *T2*, *T3*, *T4*, *T5*, *T6*, *T7*, *T8*, *T9* et *T_out* qui correspondent respectivement aux lumières et aux températures dans les différentes pièces de la maison et à l'extérieur. Ainsi que *RH_1*, *RH_2*, *RH_3*, *RH_4*, *RH_5*, *RH_6*, *RH_7*, *RH_8*, *RH_9* et *RH_out* : les valeurs d'humidité dans les pièces et à l'extérieur. Mais également des variables numériques continues : *Press_mm_hg*, *Windspeed*, *Visibility*, *Tdewpoint*, *rv1*, *rv2*, *NSM* respectivement la pression, la vitesse du vent, la visibilité, 2 variables aléatoires et le nombre de secondes depuis minuit. Ensuite, nous avons également des variables qualitatives comme *WeekStatus*, *Day_of_Week*, *DayType*, *InstantF* et *Month* mais aussi *Heure* et *Instant* (sous forme continue et numérique cette fois).

1. elles sont disponibles sur le site : <https://www.kaggle.com/competitions/https-www-kaggle-com-c-home-energy/data>

2.1 Etat général

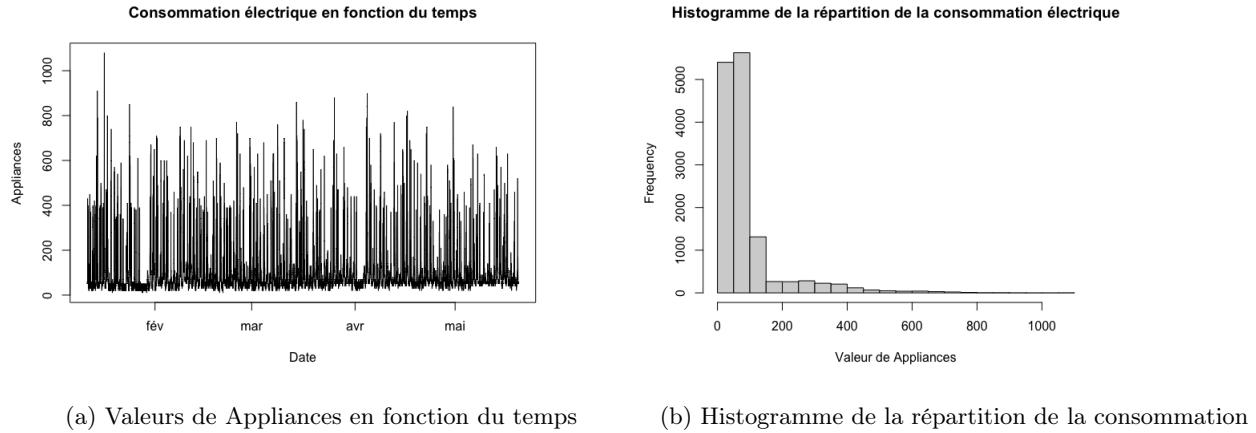


FIGURE 1 – Graphique des valeurs de Appliances

On peut voir sur la figure 1a que l'on a beaucoup de pics, on peut assez facilement supposer que cela correspond à l'utilisation de l'électroménager non quotidien et qui consomme beaucoup d'électricité comme par exemple le four, la machine à laver etc. On voit aussi que l'on a un petit fond de consommation : probablement dû aux consommations permanentes telles que le réfrigérateur ou la VMC. On remarque également sur cette même figure des périodes de "creux" d'environ 1 semaine en février et avril. On peut supposer que ce sont des périodes où les habitants de la maison étaient partis en vacances et donc n'utilisaient plus leur électroménager.

On voit sur cette figure beaucoup d'oscillations dans les valeurs d'Appliances. Cela est notamment dû au fait du bruit important de nos données mais aussi de paramètres que nous ne connaissons pas forcément. D'autre part le fait de n'avoir que 5 mois de données limite l'exactitude de la prédiction. En effet, on peut faire l'hypothèse que d'une année sur l'autre les habitudes de consommation des familles ne changent pas radicalement. On aurait donc pu éventuellement essayer de préciser nos prédictions à l'aide de séries temporelles.

Intuitivement on se dit que la consommation doit beaucoup dépendre de l'horaire et du jour : on ne consomme pas autant le mardi entre 1h et 2h du matin que lors du dîner du samedi soir. La figure 2, confirme nos dires et nous montre l'importance des variables *Instant* - *InstantF* et *Day_of_week* - *DayType*. En effet, on aperçoit sur cette figure la moyenne par heure d' *Appliances* selon le jour de la semaine. On voit que ce profil de consommation moyen varie énormément selon les jours de la semaine. Cela est sûrement dû aux activités des habitants de la maison. On peut par exemple supposer qu'ils sont en télétravail le lundi et que donc la consommation est plus haute etc. Cette étude témoigne de l'importance de considérer le jour en tant que facteur pour certaines de nos variables, dont *Instant* (c'est ce que l'on verra en autre avec *gam* dans la partie prévision).

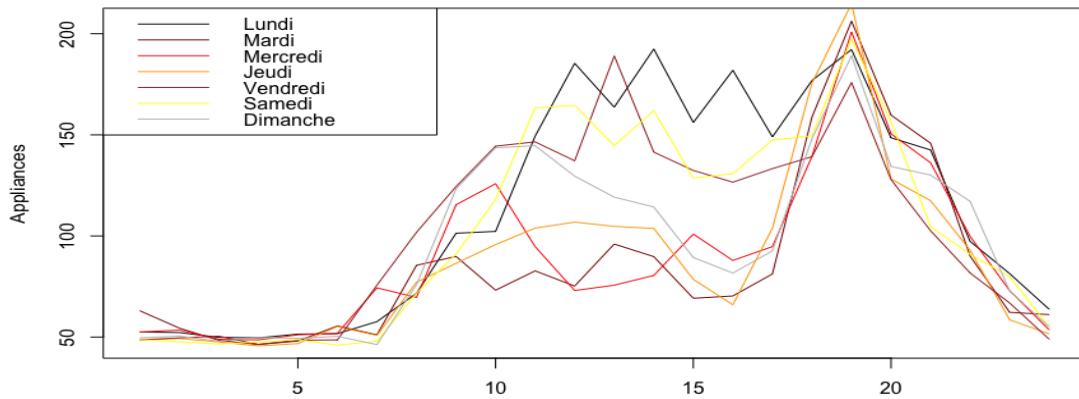


FIGURE 2 – Profil de consommation moyen selon les jours de la semaine

2.2 Corrélations entre variables

Nous avons lors de l’analyse exploratoire, chercher à identifier des corrélations entre les variables. Cela nous permettait d’une part de commencer à trouver les variables les plus explicatives pour déterminer *Appliances* mais aussi à identifier des variables explicatives fortement corrélées entre elles afin d’éventuellement les éliminer pour éviter le sur-apprentissage.

On peut voir sur le `corrplot` de la figure 3 que certaines données étaient effectivement très corrélées entre elles, en particulier les températures et humidités intérieures. On voit également une corrélation très proche de 1 pour les couples *NSM* et *Instant*, *BE_load_actual_entsoe_transparency* et *BE_load_forecast_entsoe_transparency*, *BE_wind_onshore_generation_actual* et *BE_wind_onshore_profile*, ce qui s’explique facilement d’un point de vue théorique.

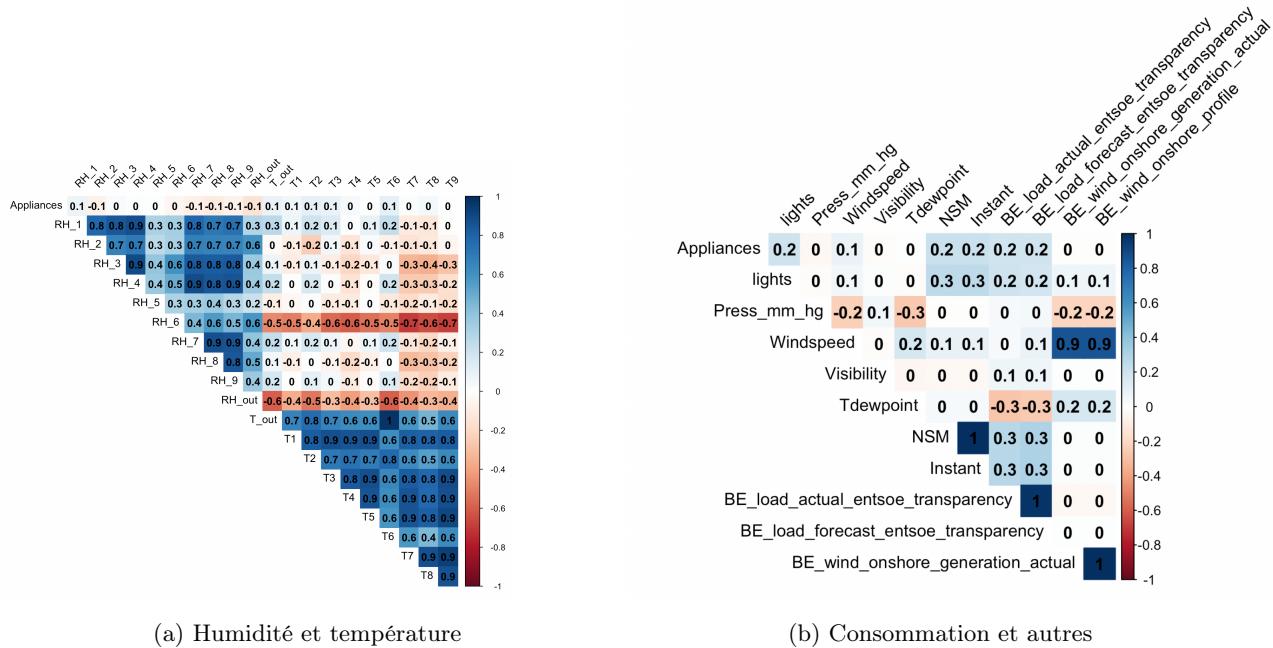


FIGURE 3 – Plot des corrélations pour Appliances et d’autres variables

2.3 Les caractéristiques des valeurs extrêmes

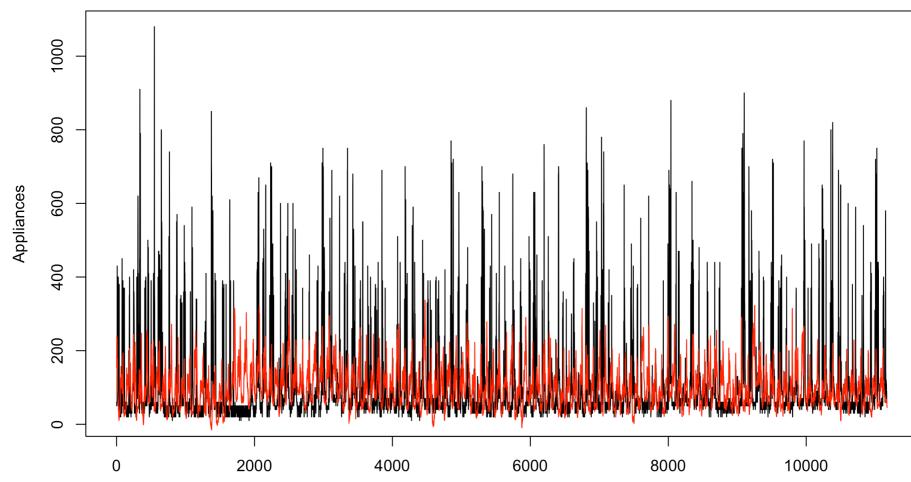


FIGURE 4 – Valeurs prédites avec la régression linéaire stepwise AIC et valeurs réelles de *Appliances*

Nous avons remarqué que nous prédisions assez mal les valeurs extrêmes, comme cela est illustré en figure 4 avec la régression linéaire stepwise AIC². En conséquence, nous avons été regarder si certains paramètres semblaient être particulièrement corrélés aux valeurs extrêmes de *Appliances* (sans forcément

2. Décrise dans le paragraphe 3.2

jouer un rôle important lorsque la consommation n'est pas extrême).

meannonext - meanext	-668.042	-5.297	RH_8	0.6547717
Appliances	-668.0423		T9	0.2110093
lights	-5.296813		RH_9	-0.2610512
T1	0.09799478		T_out	-0.4026895
RH_1	-1.484019		Press_mm hg	-0.2688664
T2	-0.3023728		RH_out	5.283093
RH_2	0.1937365		Windspeed	-0.9256145
T3	-0.4194756		Visibility	0.8813945
RH_3	-0.6196979		Tdewpoint	0.4204967
T4	-0.02054954		rv1	-2.161314
RH_4	-0.6695348		rv2	-2.161314
T5	0.3150501		NSM	-8329.673
RH_5	1.938274		Instant	-4.627596
T6	-0.7670423		Posan	0.02541782
RH_6	4.79586		BE_load_actual_ents...	-916.1171
T7	0.05217992		BE_load_forecast_ent...	-834.4824
RH_7	0.4988385		BE_wind_onshore_ca...	0
T8	0.08222552		BE_wind_onshore_ge...	-4.629796
			BE_wind_onshore_pr...	-0.003710487

FIGURE 5 – Moyenne des variables explicatives numériques des valeurs non extrêmes ($Appliances \leq 700$) - celle des valeurs extrêmes

La figure 5 montre que la moyenne de $BE_load_actual_entsoe_transparency$ et celle de $BE_load_forecast_entsoe_transparency$ sont significativement différentes quand $Appliances$ prends des valeurs extrêmes. Ce sont donc des variables qu'il peut être intéressant de "transformer" par exemple sur une base de splines (cela sera discuté dans le paragraphe 3.3.4).

3 Prévisions

La première partie de notre prévision a reposé directement sur l'analyse descriptive. En effet, on savait que l'on voulait éviter d'avoir trop de variables dans notre modèle d'autant plus si elles ne portaient pas ou peu d'information. Nous avons donc directement éliminé les variables *BE_ wind_onshore_capacity*, *Posan* et *Day_of_week*. Les 2 premières car elles prenaient toujours la même valeur et la seconde car elle portait exactement la même information que *DayType*. De la même manière nous voulions éliminer *WeekStatus* qui contenait 2 modalités : *WeekDay* et *Weekend* des informations implicitement détenues dans les 7 modalités que prends *DayType*. Nous avons aussi voulu éliminer pour chaque couple mentionné dans la section corrélation une des deux variables , toujours avec l'hypothèse sous-jacente que la seconde de la paire n'apporterait pas plus d'informations que la première. Ensuite, il y avait le quid de *Instant* et *InstantF* qui portent la même information mais d'une manière différente : *Instant* est un nombre décimal $\in [0, 48[$ correspondant à l'heure et minutes de la journée de manière "continue" tandis que *InstantF* est une heure sous le format h:m donc utilisée comme facteur. Notre 1ère envie a été de privilégier *Instant*, et de mettre de côté *InstantF*, car elle permettait plus facilement de regrouper des données proche, mais il restait le problème à minuit d'une part (on passait de 24 à 0). Et d'autre part, après des tests sur nos forêts aléatoires nous avons vu que *InstantF* était également importante afin de prédire *Appliances*. Cela réside dans la considération en tant que facteur d'*InstantF* : lors de la formation des arbres on peut alors regrouper différentes heures où la consommation est globalement similaire (par exemple 23h et 7h) alors que celles-ci ne pourraient pas avoir été regroupées ensemble en 1 seule coupure dans notre arbre avec *Instant* car elles prennent des valeurs numériques très différentes. Néanmoins le fait de considérer *Instant* comme valeur numérique permet de mieux regrouper les heures "proches". Nous avons aussi choisi de ne pas tenir compte de *rv1* et *rv2* car ce sont simplement des variables aléatoires et donc d'un point de vue théorique elles semblent peu pertinentes pour nous aider dans la prédiction de *Appliances*. Cela s'est confirmé lors de l'observation de l'importance des variables sur nos premiers modèles de forêts aléatoires (voir paragraphe 3.3.2)

3.1 Objectifs et critères de validation

3.1.1 Erreurs

— Les échantillons de test

Afin de calculer notre erreur de prédiction, nous avons utilisé plusieurs échantillons de test. La taille de chacun de ces échantillons correspondait à 20 % de notre set d'apprentissage, mais la façon dont ces 20% étaient répartis différait selon les échantillons. Le premier était composé de valeurs tirées aléatoirement dans notre jeu de données : nommé "eval_sample", le problème d'un échantillon tel que celui-ci est qu'il ne nous informe peu sur notre capacité réelle à prédire le futur. En effet, l'on test sur des données appartenant à l'ensemble du data set. C'est pourquoi nous avons également utilisé un second échantillon cette fois-ci composé uniquement des données les plus récentes (i.e.

la fin de notre jeu de données) qui se montre plus pertinent lorsque l'on cherche à effectuer des prédictions sur le futur. Néanmoins, comme le test de set utilisé sur Kaggle était composé de 30% des données de la fin du dataset et 70% de données tirées aléatoirement, nous avons mis au point un 3ème échantillon de test avec les mêmes proportions. C'est celui que l'on utilisera par la suite pour comparer les différents modèles.

— Les calculs d'erreur utilisés

— L'erreur quadratique moyenne (MSE)

C'est la principale erreur que nous utiliserons pour comparer nos différents modèles entre eux.

Elle correspond à la moyenne des carrés des erreurs de l'estimateur : $\|\hat{\theta} - \theta\|^2$ et le RMSE qui est la racine de cette même entité.

— Le R^2 et R^2 ajusté

Premièrement le R^2 est l'erreur quadratique totale (somme des carrés des erreurs). Tandis que le R^2 ajusté est une modification du R^2 tenant compte du nombre de variables prédictives dans le modèle. Ce score augmente que si la nouvelle variable améliore le modèle plus que ce à quoi l'on pourrait s'attendre en tirant au hasard.

— L'erreur de validation croisée (CV)

Très utile pour calculer des erreurs de prévision dans le cas de données iid, nous l'avons utilisé dans certains modèles où l'on ne prenait pas en compte le temps. On l'a calculé de la manière suivante : d'abord $\forall i \in 1, \dots, N$, on a $\hat{y}_{-i} \sum_{j \neq i} l(y_j, \mu(x_j))$ avec l la fonction de perte. Puis, $CV(\hat{y}) = \sum_i (l(y_i, \hat{y}_{-i}))$

— L'AIC et le BIC

L'AIC de son vrai nom "Critère d'information de Akaike" est une erreur pénalisant le surapprentissage elle est calculée par la formule : $AIC = 2k - 2 \ln(L)$ avec k le nombre de paramètres et L le maximum de la fonction de vraisemblance du modèle. Et BIC "Critère d'information bayésien" qui a la même volonté mais où la pénalisation dépend de la taille N de l'échantillon : $BIC = -2 \ln(L) + k \ln(N)$.

— Le bootstrapping pour stabiliser nos erreurs

Afin d'obtenir des résultats stables, nous avons utilisé du bootstrapping pour tester nos modèles. En effet, avec les mêmes variables en entrée et les mêmes données d'entraînement, les prévisions changeaient d'une compilation à l'autre dû à une part d'aléatoire non contrôlable dans nos modèles. En conséquence, les scores d'erreurs (rmse entre autres) prenaient des valeurs différentes (parfois jusqu'à deux unités de différence), et rendaient la comparaison entre les différents modèles difficile et souvent fausse. Pour palier à ce problème, nous avons donc créé des fonctions permettant d'effectuer plusieurs prévisions et de moyennner le score d'erreur obtenu pour chacune d'entre elles. De ce fait, nous pouvions tirer des conclusions plus réalistes sur la justesse de nos modèles. Néanmoins, cela a eu l'inconvénient d'allonger considérablement le temps de calcul. Donc nous avons choisi un nombre de répétition plus ou moins petit selon la finalité et l'utilisation de l'erreur, généralement 5 répétitions pour une première approche et plus lorsque l'on souhaitait vraiment comparer la

performance entre 2 modèles. Cette fonction affichait également un boxplot des scores obtenus permettant de vérifier que la variance entre ses scores n'était pas trop grande.

3.2 Première approche par régression linéaire

Naturellement, on a commencé à essayer de prédire *Appliances* avec la régression linéaire, l'un des modèles les plus basiques d'apprentissage, en commençant par une approche "brute" c'est à dire avec un modèle complet : où l'on a utilisé toutes les variables à notre disposition, sans transformations (sauf les variables portant exactement la même information, pour des raisons d'inversion de matrice lors de la prédiction).

```
full <- names(Data0)[-c(1,2,32,36,37,41)]
eq <- paste0("Appliances ~", paste0(full, collapse='+'))
full.model <- lm(eq, data = Data0[-eval_sample_semi_end,])
summary(full.model)

full.model.predict<-predict(object = full.model, newdata = Data0[eval_sample_semi_end,])
rmse(y=Data0[eval_sample_semi_end,]$Appliances, ychap = full.model.predict)
```

Le rmse pour ce modèle était de 94.797, soit beaucoup moins bien que ce que l'on peut espérer au vu des résultats finaux de l'article. Une analyse plus exhaustive de summary nous dit que la statistique de Fisher est de 21.33 avec 179 et 10993 degré de liberté, qui est bien supérieur à la valeur critique pour le test de l'hypothèse nulle (toutes les variables explicatives sont nulles) ; et le R^2 ajusté est de 0.2457.

Puis, en regardant en détail la significativité de la t-value pour chaque variable, on a également vu que les variables les plus importantes semblaient être : *InstantF*, *lights*, *RH_1*, *RH_2*, *RH_3*, *RH_4*, *RH_5*, *RH_8*, *T2*, *T3*, *T8*, *T9*, *T_out*, *Windspeed*, *WeekStatus*, *DayType*, en effet elles avaient une t-value inférieure au quantile du test de Student à 5%. De plus, *Heure*, *rv2*, *Instant* n'étaient pas du tout utilisées dans le modèle confirmant nos dires précédents.

Nous avons ensuite voulu améliorer notre modèle en ne sélectionnant que les variables pertinentes. Nous avons essayé plusieurs modèles "à la main" en fonction des résultats que l'on obtenait. Puis nous avons finalement automatisé ce processus chronophage. Pour cela, nous nous sommes basé sur le critère de l'AIC qui était le plus approprié car pénalisait l'ajout d'une nouvelle variable. Et nous avons fait une sélection stepwise : le principe est qu'à chaque étape on ajoute la variable minimisant le plus l'AIC, et ce, jusqu'à atteindre le seuil où l'ajout de n'importe laquelle des variables restantes ferait augmenter l'AIC.

On voit sur la figure 6 que le modèle final ne contient plus que 25 variables, ce qui est une grosse réduction par rapport au modèle complet. Et le rmse sur le même échantillon de test est de 94.757, sensiblement équivalent à celui obtenu avec toutes les variables !

```
> step.model$anova
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
Appliances ~ lights + T1 + RH_1 + T2 + RH_2 + T3 + RH_3 + T4 +
RH_4 + T5 + RH_5 + T6 + RH_6 + T7 + RH_7 + T8 + RH_8 + T9 +
RH_9 + T_out + Press_mm hg + RH_out + Windspeed + Visibility +
Tdewpoint + rv1 + rv2 + NSM + WeekStatus + DayType + InstantF +
Instant + BE_load_actual_entsoe_transparency + BE_load_forecast_entsoe_transparency +
BE_wind_onshore_generation_actual + BE_wind_onshore_profile

Final Model:
Appliances ~ lights + T1 + RH_1 + T2 + RH_2 + T3 + RH_3 + RH_4 +
T5 + RH_5 + T6 + RH_6 + T7 + RH_7 + T8 + RH_8 + T9 + RH_9 +
T_out + Windspeed + Tdewpoint + DayType + InstantF + BE_wind_onshore_profile
```

FIGURE 6 – Sommaire de la régression stepwise prenant pour critère l’AIC

Nous avons ensuite essayé d’améliorer notre régression linéaire avec l’aide de `glmnet`, qui offre des techniques d’optimisation efficaces pour la régression logistique régularisée. Nous avons donc testé la sélection Lasso (contraintes sur la norme 1, $\alpha=1$) et Ridge (contrainte sur la norme 2, $\alpha=0$) puis Elasticnet (avec $\alpha=0.5$) permettant de combiner les avantages des 2 sélections précédentes. Néanmoins, nous sommes vite arrivés à la conclusion que la regression linéaire n’était pas la technique la plus performante, et il était difficile de faire baisser les scores rmse obtenus jusqu’ici. C’est pourquoi nous nous sommes tournés vers les forêts aléatoires.

3.3 Forêts aléatoires

3.3.1 Principe

Les forêts aléatoires reposent sur un algorithme d’apprentissage automatique consistant à faire tourner en parallèle plusieurs centaines d’arbres de décision construits aléatoirement. Ces arbres sont construits avec à chaque noeud un choix parmi $q \in [1, p]$ variables, p étant le nombre total de variables du modèle (Bagging). Cela permet de construire des arbres décorrélés et donc lorsqu’ensuite on agrège les prédicteurs obtenus pour en former un plus robuste, on réduit la variance du problème.

Les forêts aléatoires disposent de plusieurs autres avantages. Premièrement il n’y a peu de problème de sur-apprentissage (c’est pour cela que l’on utilise pas des erreurs telles que l’AIC et/ou le BIC dans cette section). Deuxièmement, leur performance est meilleure que les arbres de décision. Et ensuite, cette technique ne nécessite pas non plus de regarder la validation croisée grâce aux échantillons aléatoires "Out of bag".

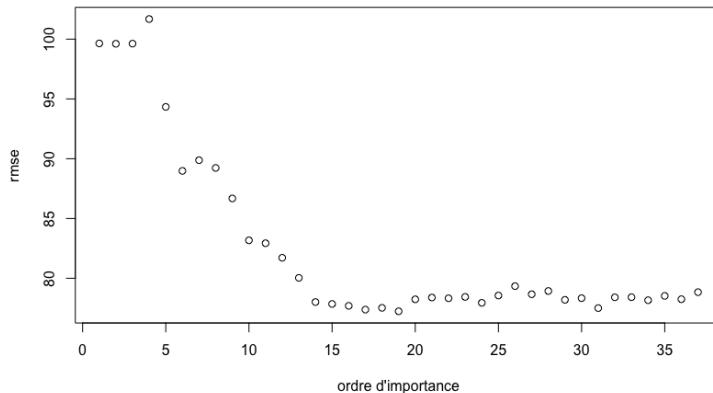
3.3.2 Premiers modèles

Dans un premier temps nous avons travaillé sur le modèle complet, c'est à dire : `ranger(Appliances ~ lights + T1 + RH_1 + T2 + RH_2 + T3 + RH_3 + T4 + RH_4 + T5 + RH_5 + T6 + RH_6 + T7 + RH_7 + T8 + RH_8 + T9 + RH_9 + T_out + Press_mm hg + RH_out + Windspeed + Visibility + Tdewpoint + rv1 + rv2 + NSM + WeekStatus + Day_of_week + DayType + InstantF + Instant + Heure + BE_load_actual_entsoe_transparency + BE_load_forecast_entsoe_transparency + BE_wind_onshore_generation_actual + BE_wind_onshore_profile, data=Data0[-eval_sample_semi_end,], importance ="permutation")`.

Le rmse de ce modèle était de l'ordre de 78.3, beaucoup mieux que ce que l'on obtenait avec la régression linéaire.

InstantF	NSM	Instant
4817.11630	4724.28359	4391.83638
RH_3	T3 BE_load_forecast_entsoe_transparency	3023.94973
3910.18163	3214.25711	RH_1
Heure	BE_load_actual_entsoe_transparency	2520.84463
2976.84858	2724.47989	RH_8
T2	T8	2133.17657
2395.03039	2207.84790	T6
RH_2	Day_of_week	2016.97459
2069.74911	2037.67948	RH_7
T_out	RH_4	1917.12935
2008.25578	1997.27614	T4
T9	T5	1815.99127
1858.31703	1835.60060	RH_9
RH_6	Tdewpoint	1695.92045
1808.87709	1727.60618	RH_out
T7	Press_mm hg	1568.27091
1689.00360	1651.36314	RH_5
T1	DayType	1224.07455
1534.47650	1474.47948	BE_wind_onshore_profile
Windspeed	BE_wind_onshore_generation_actual	1035.24254
1074.13405	1047.27672	rv1
Visibility	WeekStatus	60.32180
322.88100	307.34989	
rv2		
55.38081		

(a) Valeurs numériques et ordre d'importance



(b) rmse du modèle en fonction du nombre de variable (classée par importance)

FIGURE 7 – Importance des variables dans le modèle de forêt améatoire avec toutes les variables explicatives

En regardant l'importance des variables du problème sur la figure 7a on voit par ailleurs que nos

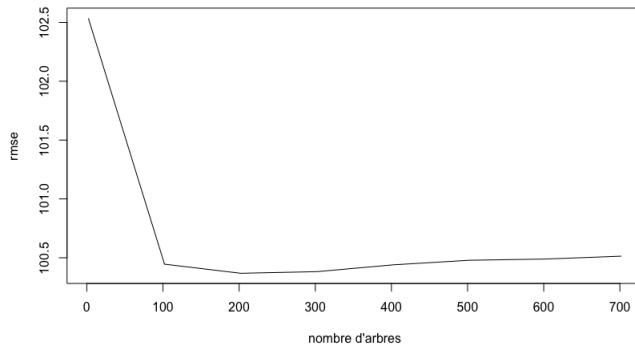
suppositions faites plus haut semblent vraies et surtout la figure 7b nous montre que les 17 variables les plus importantes portent quasiment toute l'information et que les suivantes ne sont pas forcemment pertinentes dans ce cas, voir même que leur ajout semblent faire augmenter légèrement notre erreur.

En ne gardant que les 17 plus importantes variables, nous obtenons le modèle (3) :

```
ranger(Appliances ~ RH_1 + T2 + RH_2 + T3 + RH_3 + RH_4 + T6 + T7 + T8 + RH_8 + T_out +
NSM + Day_of_week + InstantF + Instant + Heure + lights+ BE_load_actual_entsoe_transparency +
BE_load_forecast_entsoe_transparency, data=Data0[-eval_sample_semi_end,], importance = "permutation")
```

3.3.3 Optimisation des paramètres

Comme les forêts aléatoires sont construites à partir d'un certains nombres de paramètres tels que la profondeur des feuilles, le nombre de variables aléatoires tirées à chaque coupure ou encore, le nombre d'arbres dans chaque forêt. Il est intéressant d'aller regarder quels sont leurs impacts sur la justesse de nos prédictions.



```
rf0 <- ranger(Appliances ~ lights + T1 + RH_1 + TZ + RH_2 + T3 + RH_3 + T4 + RH_4 + T5 + RH_5 + T6 + RH_6 + T7 + RH_7 + T8 + RH_8 + T9
                data=Data0[-eval_sample, ], importance='permutation', oob.error=T, mtry=10, max.depth=1, num.trees = 800)

rf0.trees <- predict(rf0, predict.all=T, data=Data0[eval_sample, ])$predictions

rmse_tree <- lapply(seq(2, 800, by=100), function(x){rmse(y=Data0[eval_sample, ]$Appliances, yhat=rowMeans(rf0.trees[,1:x]))} )

plot(seq(2, 800, by=100), rmse_tree%>%unlist, type='l', xlab="nombre d'arbres", ylab="rmse")
```

FIGURE 8 – Optimisation du nombre d'arbre dans chaque forêt aléatoire

La première étape de l'optimisation a été d'étudier le nombre d'arbre à calculer par forêt aléatoire, en effet bien que cela ne permettait pas d'aminoindrir les erreurs, c'était un gain de temps considérable. Nous avons donc regardé les performances de `ranger` selon le nombre d'arbre (figure 8) . On y voit qu'à partir de 200 arbres, le résultat du rmse ne changeait pas sensiblement, voir réaugmentait. Nous avons donc choisi

pour la suite de prendre num.trees=200 au lieu de 500 qui est le paramètre par défaut de la fonction. C'est aussi un gain de temps pour l'exécution.

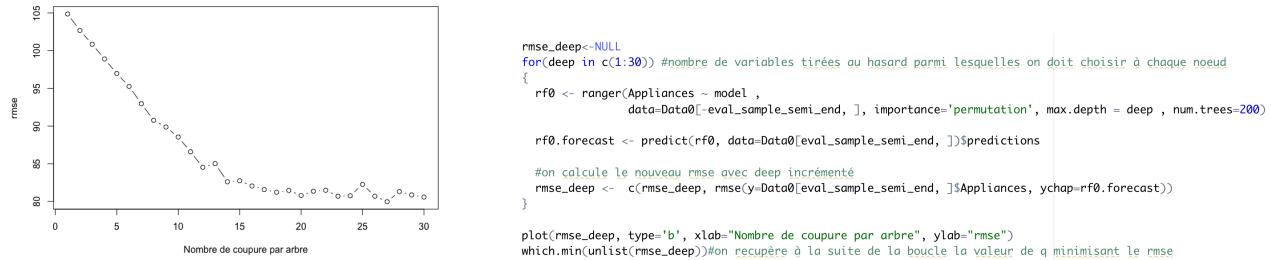


FIGURE 9 – Optimisation du nombre de coupure dans chaque arbre
Les résultats ont été obtenus à partir du modèle (3)

Ensuite, nous avons voulu optimiser, toujours dans cette optique de gain de temps de calcul, la profondeur des arbres. C'est à dire le nombre de coupures effectuées par arbre. Sur la figure 9, on peut apercevoir qu'à partir de 18 coupures, le gain en terme d'erreur est peu important. Donc, pour tester les différents modèles nous avons utilisé 18 coupures par arbre seulement.

```

rmse_mtry <- NULL
for(mtry in c(1:10)) #nombre de variables tirées au hasard parmi lesquelles on doit choisir à chaque noeud
{
  rf0 <- ranger(Appliances ~ model, data=Data0[-eval_sample, ], importance='permutation', oob.error=T, mtry=mtry, num.trees=200)
  rf0.forecast <- predict(rf0, data=Data0[eval_sample, ])$predictions

  rmse_mtry <- c(rmse_mtry, rmse(y=Data0[eval_sample, ]$Appliances, yhat=rf0.forecast))#on calcule le nouveau rmse avec mtry incrémenté
  print(mtry)
}

plot(rmse_mtry, type='b')
which.min(unlist(rmse_mtry))#on récupère à la suite de la boucle la valeur de q minimisant le rmse
  
```

FIGURE 10 – Code R pour optimiser le nombre de variables aléatoires tirées à chaque noeud

Finalement, nous avons optimisé pour chacune des solutions envisagées le nombre de variables q tirées à chaque noeud. Nous voulions que ce nombre soit assez petit devant p pour obtenir des arbres décorrélés et donc une variance faible de l'estimateur mais qui minimise le rmse (donc avec un biais pas trop important). Nous avons donc utilisé le code en figure 10 pour pouvoir déterminer quelle valeur de q était la plus judicieuse dans chaque modèle.

3.3.4 Combinaisons de variables grâce à GAM et CART pour améliorer les performances des forêts

Une fois l'optimisation des paramètres des forêts aléatoires, le seul moyen qui pouvait permettre d'abaisser encore nos erreurs de prédiction était de jouer sur les variables en entrée. Il est par exemple possible de faire des transformations, notamment par des fonctions polynomiales par morceaux , ou des fonctions impliquant 2 variables qui interagissent entre elles etc.

a) Intéractions entre variables

Afin de déterminer les interactions entre les variables, nous avons observé l'arbre de décision du modèle complet. Pour cela nous avons utilisé la fonction R `rpart`.

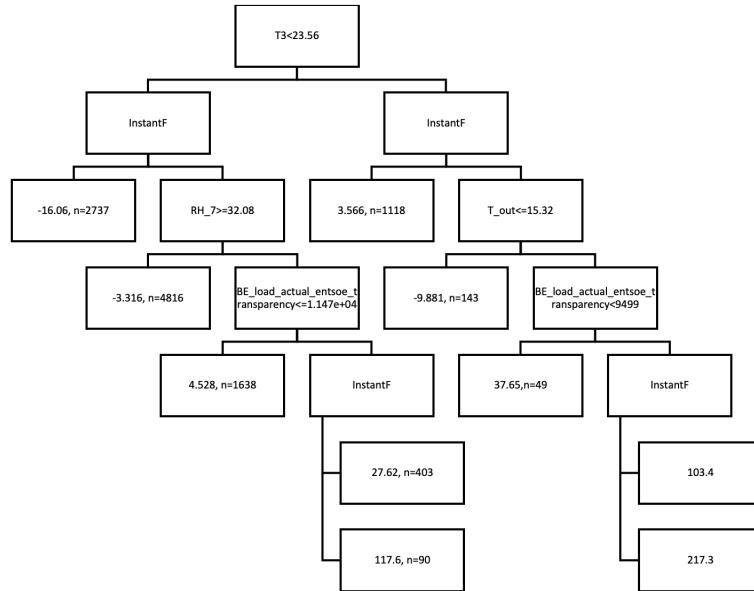


FIGURE 11 – Premiers noeuds de l'arbre de décision obtenu avec l'ensemble des variables explicatives non redondantes

`rpart` peut également être directement utilisée comme technique de prévision, toutefois elle est beaucoup moins efficace que les forêts aléatoires et présente souvent un estimateur avec une variance importante. Par exemple le rmse pour cette technique avec le modèle complet (et l'échantillon test semi_end) est de 93.17, tandis qu'il est de l'ordre de 79 avec les random forests. Il n'a donc pas été exploité pour la prévision en tant que telle. Néanmoins, il nous a servit à essayer de détecter des interactions entre certaines variables afin de les exploiter ensuite avec `gam`, puis de les injecter dans nos forêts aléatoires.

On peut voir sur le graphe de la figure 11 qu'il semble y avoir des interactions entre *InstantF* et *T3* ou entre *T_out* et *BE_load_actual_entsoe_transparency*, mais encore *BE_load_actual_entsoe_transparency* et *RH_7* etc. Ces interactions peuvent indiquer qu'une transformation impliquant les 2 moitiés du couple de variables en interactions est peut-être plus pertinent pour notre modèle. C'est ce qui est exploité dans le paragraphe suivant.

b) Transformation des variables avec des bases de splines

Les bases de splines nous ont servi à améliorer nos prédictions de 2 manières différentes. La première,

en cherchant des transformations non linéaires de variables seules permettant de mieux expliquer *Appliances*. Et la seconde en cherchant des transformations non linéaires impliquant 2 variables.

Nous avons donc commencé par chercher les transformations impliquant 2 variables en interaction. Le premier constat étant la difficulté et l'inefficacité d'une régression sur base de splines pour les intercations avec *InstantF*. Cela est dû au nombre de niveau que contient *InstantF* (52) qui implique que pour chaque transformation avec en facteur *InstantF*, on a 52 nouvelles variables à prendre en compte dans le modèle.

```
g3 <- gam(Appliances~s(RH_7, BE_load_actual_entsoe_transparency, k=20), data=Data0[-eval_sample_semi_end, ])
summary(g3)
g3.forecast <- predict(g3, newdata=Data0[eval_sample_semi_end, ])
rmse(y=Data0[eval_sample_semi_end, ]$Appliances, ychap=g3.forecast)
#plot(g3)
eq<-g3$residuals ~ NSM + InstantF + Instant + RH_3 + T3 + Heure + BE_load_forecast_entsoe_transparency +
  RH_1 + T2 + BE_load_actual_entsoe_transparency + T9 + T8 +
  RH_2 + RH_8 + T6 + RH_4 + T5 + T_out + RH_6 +DayType

rf_res <- ranger(eq, data=Data0[-eval_sample_semi_end, ], importance='permutation')
rf_res.forecast <- predict(rf_res, data=Data0[eval_sample_semi_end, ])$predictions
rmse(y=Data0[eval_sample_semi_end, ]$Appliances, ychap=g3.forecast+rf_res.forecast)
```

FIGURE 12 – Code R pour la régression sur base de splines de *RH_7* et *BE_load_actual_entsoe_transparency*

La plupart des essais ont été infructueux. Il est difficile d'identifier clairement des interactions entre variables. Néanmoins, celle liant *RH_7* et *BE_load_actual_entsoe_transparency* a bien fonctionnée et nous a permis de gagner quelques dixième sur le score, sans pour autant augmenter le nombre de variable (nous avons pu retirer *RH_7* du modèle). Le code est visible en figure 12. On y a choisi $k=20$ au vu de l'analyse du **summary**, il y était indiqué que le degré de liberté estimé était de l'ordre de 16. Choisir un k plus petit aurait donc conduit à l'obtention d'un minimum plus grand et il n'y avait pas d'intérêt non plus à prendre un k trop grand. Or, bien que performant à l'entraînement ce modèle était très mauvais sur le test kaggle. Nous l'avons donc abandonné. La seule combinaison trouvée nous ayant permis d'améliorer notre score est `gam(Appliances ~ s(Instant, k=20, by = as.factor(DayType)), data = Data0[-eval_sample_semi_end,])` en l'insérant ensuite dans une forêt aléatoire. Cette interaction avait été remarquée à la phase d'analyse exploratoire où l'on avait vu que selon le jour et l'instant la consommation d'*Appliances* différait. Nous avons donc créé 7 nouveaux termes d'*Instant* (un pour chaque jour de la semaine). Et après optimisation du paramètre `mtry`, l'erreur rmse a été abaissée à 77.8 en moyenne.

Nous avons ensuite essayé des transformations de variables seules. En particulier sur *BE_load_actual_entsoe_transparency* et *BE_load_forecast_entsoe_transparency* dont de grandes valeurs, nous l'avons vu plus haut, étaient souvent liées à des consommations plus importantes de *Appliances*. Associée, au modèle précédent, cette conclusion s'est avérée concluante. Nous l'avons donc intégrée dans notre modèle.

Pour les autres variables que l'on a essayé de transformer via `gam`, bien que certaines soient montrées performantes à l'entraînement, la plupart se sont révélées médiocres à l'issue du test sur la compétition. Cela

est sans-doute dû à une forte dépendance de ces techniques aux données d'entraînement et donc une tendance à avoir un modèle qui sur-apprends.

4 Conclusion

Pour conclure, notre modèle final est donc : `lights + T2 + T3 + BE_load_forecast_entsoe_transparency + T6 + T7 + T8 + T_out + RH_1 + RH_2 + RH_3 + RH_4 + RH_7 + RH_8 + RH_9 + RH_out + DayType + InstantF + BE_load_actual_entsoe_transparency + gterm_BE_laet + gterm1Instant + gterm2Instant + gterm3Instant + gterm4Instant + gterm5Instant + gterm6Instant + gterm7Instant`. Il nous a permis d'obtenir un score de 78.897 sur la compétition kaggle. Ce n'est pas le meilleur score que l'on a obtenu : on atteignait 78.687 avec le modèle : `lights + T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + T9 + T_out + RH_1 + RH_2 + RH_3 + RH_4 + RH_5 + RH_6 + RH_7 + RH_8 + RH_9 + RH_out + Press_mm hg + Windspeed + Tdewpoint + Visibility + Day_of_week + Instant + BE_load_actual_entsoe_transparency + BE_wind_onshore_profile + BE_load_actual_entsoe_transparency** 2 + BE_load_forecast_entsoe_transparency`. Néanmoins ce modèle contenait beaucoup plus de variables, et comme le gain en terme d'erreur était très minime, nous avons préféré garder l'autre modèle.

Nous avons fait état dans ce rapport état d'une petite partie des modèles créés et testés, mais beaucoup d'autres n'ont pas pu être exposés, dont la plupart se sont révélés peu concluants. Bien sûr il est certain que nous pourrions encore essayer d'améliorer les modèles présentés. Par exemple, comme l'on a vu que beaucoup des variables de températures étaient corrélées on aurait pu imaginer de créer une nouvelle variable dans notre tableau étant combinaison de plusieurs variables de températures (cette nouvelle valeur aurait pu être obtenue à partir de régression linéaire par exemple). Néanmoins l'implémentation de ces modèles et les tests à effectuer dessus sont très coûteux en terme de temps et de calcul.

Par ailleurs, nous avons pu remarquer l'intérêt de l'utilisation de forêts aléatoires dans certains modèles de prévision, dont le nôtre. Ces techniques non paramétriques étaient dans notre cas beaucoup plus performantes. Toutefois elles ont le gros désavantage d'être souvent des "boîtes noires" où il est difficile d'aller carabistouiller ce qu'il se passe vraiment dedans. Cela limite donc les améliorations possibles sur ce genre de techniques.

Notre score peut sembler assez élevé par rapport à certains problèmes où les erreurs de prédictions sont beaucoup plus petites, mais au vu du bruit de nos données, on peut supposer qu'il sera difficile de faire beaucoup mieux que ce que nous avons déjà. On pourrait espérer gagner une ou deux unités sur le score dans le meilleur des cas.

Un autre axe d'amélioration aurait été de tester à chaque fois nos modèles sur plusieurs échantillons aléatoires "eval_sample" et "eval_sample_semi_end". Mais cela impliquait de faire de faire une double

boucle pour les tests d'erreurs (boucle sur les échantillons et boucle de bootstrapping) qui donnait des temps de calcul astronomiques. C'est pourquoi dans la plupart des cas nous fixions une graine pour nos échantillons aléatoires, même si du coup on peut avoir tendance à créer des modèles qui vont coller un peu trop à nos données.