Load Forecasting during the COVID Period

Halvard Bariller & Ángel Reyero Lobo *Université Paris Saclay - M1 MIA*

This document provides a detailed explanation of the various research methods that have been considered for forecasting the daily electricity load in France during the COVID period (April 2020 to January 2021).

Keywords: forecasting, electricity, machine learning methods, time series

Contents

Introduction

The main difficulty in this project is that the modeling and the predictions expected occur during a very peculiar period : the COVID crisis. No historical data could have helped encapsulate the sudden variations in electricity load we faced. We thus divided our approach into two phases

: the first one was to learn from historical data in "regular periods", thus modeling the usual electricity load, and once that done, we explored various techniques (online learning, ARIMA, experts aggregation, adaptative models...) to readjust the model to the brutal change in trend we faced in these extraordinary conditions.

In this project, the objective has been to get the best performance through a good expert aggregation. To do so, we have tried to diversify our experts as much as possible.

We have used diverse datasets to construct our models thanks to methods such as Random Forests where we do not construct the experts with the whole dataset but only with some subsets. We have also tried rolling-window methods with different sizes of window, differently weighted training examples... Hence, we managed to create training diversity with different partitioning and subsets.

Moreover, we have diversified our loss functions. Instead of only using the L2 loss to train the methods as it may be intuitive (because at the end, our objective is to minimize this cost function over the test set), we have used quantile loss (for example through the qgam). To choose the quantile where we wanted to minimize this loss, we have observed that during the first weeks of the Covid, the Load was sharply reduced. As our test set is focused on the Covid period, to improve our performances, we have tried to minimize the quantile loss over quantiles between 0.3 and 0.5. Furthermore, we have also modified the weights of the loss function to give more importance to some instances that we have considered, which could be seen as a loss change. Finally, we have also introduced a L1 loss for variable selection.

At last, we have also combined multiple methods to get a variability of algorithms. With all these methods we have tried to diversify the explanatory variables as it is done in the Random Forest. Therefore, each model may be able to add some information to the global aggregation.

I. Feature Engineering

Before any model conception, we implemented some basic but needed transformations on the train set and the test set. Indeed, features were of different kinds and we needed to clean the datasets in order to build efficient models.
The main transformations concerned categorical variables : *BH*, *DLS*, *Christmas_break*, *Summer_break* and *WeekDays* were recoded as factors. Regarding *WeekDays*, we considered two formatting perspective on top of the initial one :

- recode a categorical variable *WeekDays2* regrouping 'Tuesday', 'Wednesday' and 'Thursday' as one and only factor level 'Workday'
- mix *WeekDays2* and *BH* : 5 factor levels for 'Monday', 'Workday','Friday','Saturday','Sunday' and one factor level 'BH' taking priority over the levels of *WeekDays2*

However, models performance led us to retain only the original *WeekDays* as factor, without any simplification. This choice also allowed us to train daily model (cf. II.3).

One last transformation was operated on the initial datasets, recoding the variable *Date* into a numeric version of it called *Time*.

Finally, we prepare the division of the train set into two independant subsets, *sel_a* and *sel_b*, which will be used to distinguish the COVID period on train set.

II. Models

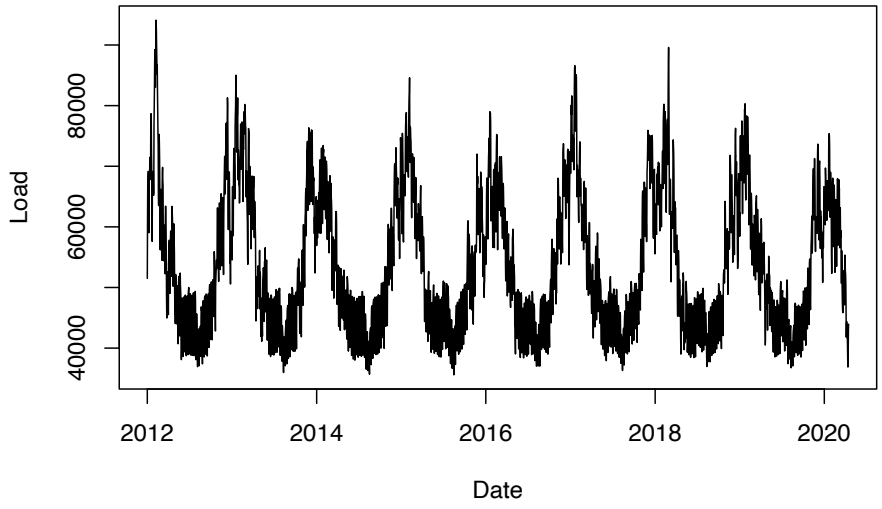
A recurrent idea throughout the approaches we have taken is the rolling-window for model training.

Models trained over huge datasets are usually better as they are able to extract more information and they avoid overfitting. In contrast, when we are treating time series, we should be able to adapt to the changes as fast as possible. Then, if we train a model with a huge dataset where only the latest data suffered a trend change, the algorithm will treat this latest data as outliers, as they are not really representative with the rest of the data. That is the case of our dataset, where the Covid definitively affected the Load trend. Therefore, to be able to learn from this information, we have added a parameter allowing to reduce the training dataset to a window size small enough to treat this period. Moreover, this mechanism allows us to introduce variables that are proper to this period such as Government Response Index, diversifying the models for the aggregation.

1. Linear model with adaptative ARIMA correction

For each of these approaches, the models are trained based on the subset of the train set anterior to COVID. The idea is to build models capturing efficiently the load variation throughout the year, independantly of any COVID effect.
We then adjust ARIMA models on the obtained residuals in order to encapsulate the previously ignored COVID effect.

```
plot(Data0$Date,Data0$Load, xlab = "Date", ylab = "Load", type='l')
```



We don't observe any obvious trend : the idea to train only of the observations anterior to 2020 seems reasonable.

a. AIC penalty on polynomial combinations

On the one hand, we have decided to combine all the variables by multiplying each variable to the others to make the model more flexible. On the other hand, this combination made the model extremely overparameterized. In order to select the variables to use in this linear model, we have applied the stepAIC to maximize the AIC criterion, which is a penalization criterion to avoid overfitting.

```
# lm_AIC <- linear_model_AIC(Data0[sel_a,])
lm_AIC <- lm(Load ~ Load.1 + Load.7 + Temp + Temp_s95 + Temp_s95_max + Temp_s99_max +
                          toy + WeekDays + BH + DLS + Summer_break + Christmas_break +
                          Time + Load.1:Temp + Load.1:Temp_s95 + Load.1:Temp_s95_max +
                          Load.1:Temp_s99_max + Load.1:toy + Load.1:WeekDays + Load.1:DLS +
                          Load.1:Summer_break + Load.1:Christmas_break + Load.7:Temp_s95_max +
                          Load.7:Temp_s99_max + Load.7:DLS + Load.7:Summer_break +
                          Load.7:Christmas_break + Temp:Temp_s95 + Temp:Temp_s99_max +
                          Temp:WeekDays + Temp:DLS + Temp:Christmas_break +
                          Temp_s95:Temp_s95_max + Temp_s95:Temp_s99_max +
                          Temp_s95:toy + Temp_s95:WeekDays +
                          Temp_s95:DLS + Temp_s95:Summer_break + Temp_s95:Christmas_break +
                          Temp_s95:Time + Temp_s95_max:Temp_s99_max + Temp_s95_max:toy +
                          Temp_s95_max:Christmas_break + Temp_s95_max:Time +
                          Temp_s99_max:toy +Temp_s99_max:WeekDays + Temp_s99_max:BH +
                          Temp_s99_max:DLS + toy:WeekDays + toy:BH + toy:DLS +
                          toy:Summer_break + toy:Christmas_break +
                          WeekDays:BH + WeekDays:DLS + WeekDays:Summer_break +
                          WeekDays:Christmas_break +
                          WeekDays:Time + BH:DLS + BH:Summer_break + BH:Christmas_break +
                          BH:Time + Summer_break:Time, data= Data0[sel_a,])
summary(lm_AIC)$adj.r.squared
```

```
## [1] 0.9949275
```

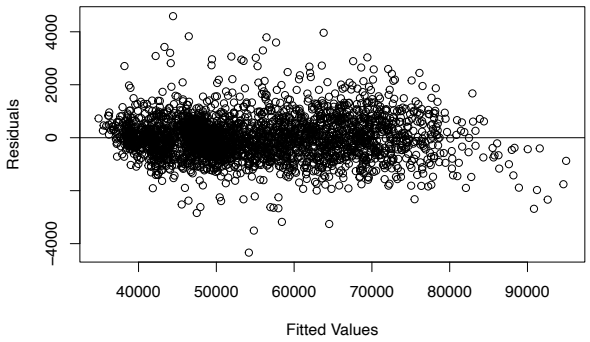
```
### CV 8-fold RMSE
rmse(lm_cvpred(Data0[sel_a,],eq = eq_lm_AIC),Data0[sel_a,]$Load)
```

```
## [1] 840
```

```
rmse(predict(lm_AIC, newdata = Data0[sel_b,]),Data0$Load[sel_b])
```

```
## [1] 1900
```

```
plot(fitted(lm_AIC), resid(lm_AIC), xlab='Fitted Values', ylab='Residuals')
abline(0,0)
```




```
library(lmtest)
#perform Breusch-Pagan test
bptest(lm_AIC)
```

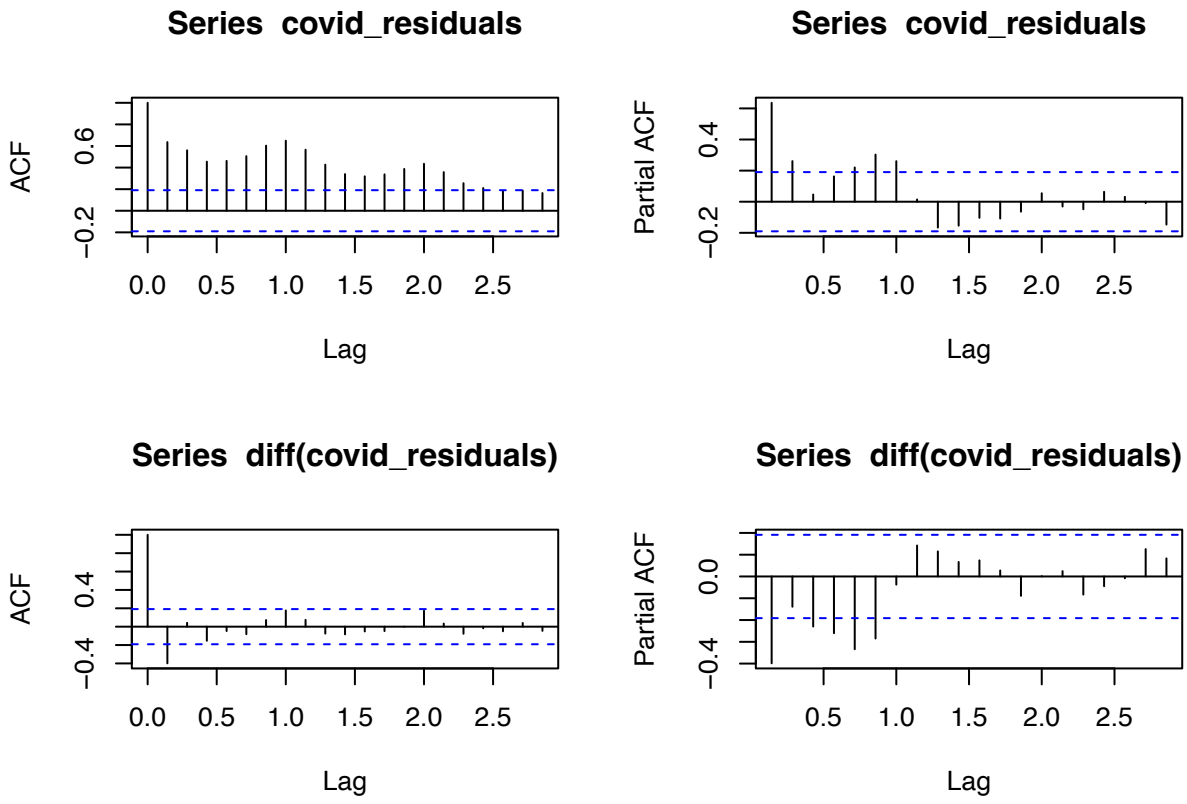
```
##
##  studentized Breusch-Pagan test
##
## data:  lm_AIC
## BP = 570.22, df = 118, p-value < 2.2e-16
```

After having computed this linear model, we noticed that the hypothesis made about the residuals homoscedasticity is not accurate. This is because we are treating a time series. To adapt this hypothesis, we introduced an Arima correction on the residuals. The heteroscedasticity of the residuals is confirmed when performing a Breusch-Pagan test on the fitted values.

We clearly see a rupture in the residuals around March 2020, corresponding to the beginning of the COVID and the first lockdown.

We have tried to find out if the residuals of the model followed any seasonality or trend. To do so, we have plotted the empiric autocorrelations and partial autocorrelations. We have also tried to use the differentiation to correct this seasonality and trend. Unluckily, all the Auto-Regressive and Moving Average algorithms that we have tried following our intuitions from this graphics were not good as the ARIMAs computed by the auto.arima minimizing the AIC. Therefore, we have just limited the methods to learn from this auto.arima.

```
covid_residuals = ts(predict(lm_AIC, newdata = Data0[sel_b,])-Data0[sel_b,]$Load,
                     frequency = 7)
par(mfrow=c(2,2))
acf(covid_residuals)
pacf(covid_residuals)
acf(diff(covid_residuals))
pacf(diff(covid_residuals))
```



b. AIC penalty on polynomial combinations with forgetting factor for adaptative model selection

We noticed that the Linear Model computes the AIC loss minimizing the addition of a loglikelihood (because we have done the gaussian hypothesis) and a variable penalty. This loglikelihood gives the same weight to all the observations. Nevertheless, as we are working with time series, we should be able to adapt the model to the new observations as fast as possible. Therefore, the latest observations should be more than the initial observations. To integrate this idea in our modeling, we have introduced a sigmoid weightening, which gives more importance to the latest observations. This will give more information about the variables as we are able to update the model variables.

c. AIC penalty on polynomial combinations for weighted linear model

We used once more the linear model selected with AIC penalty, and retrained it as a linear model attributing more weight to the observations with lower variance.

```
wt = 1 / lm(abs(lm_AIC$residuals) ~ lm_AIC$fitted.values)$fitted.values^2
weighted_lm_AIC <- lm(eq_lm_AIC, Data0[sel_a,], weights = wt)
summary(weighted_lm_AIC)$adj.r.squared
```

```
## [1] 0.9942834
```

Considering this strategy, we could also imagine a weighted regression where more weight is attributed to the training examples with the largest prediction error, in order to force a correction on the model.

This idea will be pursued with weighted GAMs and weighted Boosting models.

2. Monthly model with adaptative ARIMA correction

To pursue with this idea of leveraging the dataset diversity to improve the final expert aggregation, we trained linear models on specific time window.
The monthly model is a combination of 12 different linear models, each one fitted specifically on one month. The prediction is then made based on the month considered, calling the proper model.

3. Daily model with adaptative ARIMA correction

The same idea of time-specific model as previously detailed for monthly models has been applied here, training 7 different models, one for each weekday.

4. GAM model with adaptative ARIMA correction

We use a double penalty approach when fitting our GAMs, allowing to penalize functions in both null and range spaces.
The number of knots in GAMs has been tuned looking at the degree of freedoms with the command *gam.check*. The spline basis have been compared using cross-validation on the subset of training set prior to COVID.

a. Tuned GAM with Random Forest for variable selection

Still following the idea of diversifying our experts, we tried to diversify the explanatory variables by tuning a GAM with variable selection based on the order of importance in Random Forests constructions.

```
variable_importance <- rf_variable_selection(dataset = Data0[sel_a,],
                                             var_allowed = 13,
                                             trees = 200)
variable_importance
```

```
##                Load.1           Temp_s99_max                Load.7
##          45504069.19           14160102.52           10355011.50
##              Temp_s99               WeekDays           Temp_s95_max
##           6410920.00            6350206.01            5377583.18
##          Temp_s99_min              Temp_s95                  Temp
##           3845806.83            3071964.98            1826699.40
##          Temp_s95_min                   toy                    BH
##           1276881.34            1133271.73            1032971.23
##                 Month           Summer_break                   DLS
##           243522.13             220514.51             201261.15
##                  Date                  Time        Christmas_break
##           66478.99              60929.58              50161.85
```

```
##                     Year GovernmentResponseIndex
##                 20198.07                    0.00
```

We used the ANOVA command to perform Chi-squared test on slotted models to look for interactions between features.

```
interaction_significance(dataset = Data0[sel_a,],"Temp_s99_max","Temp_s99_min")
```

```
## [1]         NA 0.01150633
```

```
interaction_significance(dataset = Data0[sel_a,],"Temp_s95_max","Temp_s95_min")
```

```
## [1]           NA 1.202322e-11
```

```
# tuned_gam_model <- gam_tuned_model(dataset = Data0[sel_a,])
# saveRDS(tuned_gam_model, "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predictive modelling/Sa

tuned_gam_model <- readRDS(file = "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predictive model

#gam.check(tuned_gam_model)
sqrt(tuned_gam_model$gcv.ubre)
```

```
##  GCV.Cp
## 874.699
```

b. GAM with linear model features

In this scenario, we fitted a GAM using features obtained with the linear model AIC-penalized. We also added an argument allowing to consider exponentially decreasing weights on observations.

```
# gam_linear_model <- gam_lm_model(dataset = Data0[sel_a,], weighted = T)
# saveRDS(gam_linear_model, "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predictive modelling/S

gam_linear_model <- readRDS(file = "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predictive mode

#gam.check(gam_linear_model)
sqrt(gam_linear_model$gcv.ubre)
```

```
##   GCV.Cp
## 1766.294
```

The GVC/UBRE score is less encouraging as compared to the previous GAM, but just as for the linear model, the ARIMA correction worked really well on this model.

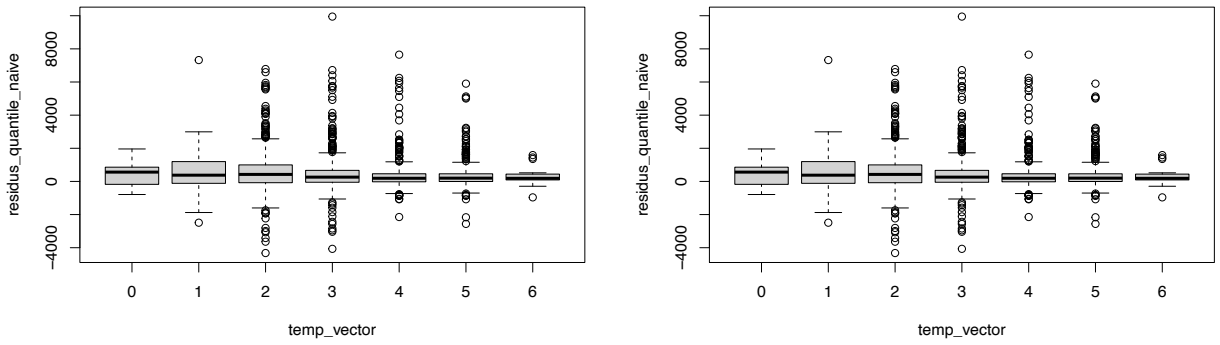
5. QGAM model with adaptative ARIMA correction

```
# naive_qgam_model <- naive_qgam_model(dataset = Data0[sel_a,],
#                                      quantile = 0.3)
#
# saveRDS(naive_qgam_model, "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predictive modelling/S

naive_qgam_model <- readRDS(file = "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predictive mode

#plot(naive_qgam_model, scale = F, page = 1)
#check(naive_qgam_model)
# check(naive_qgam_model$calibr)
```

```
residus_quantile_naive <- naive_qgam_residual_variance(dataset = Data0[sel_a,],
                                                       model = naive_qgam_model)
```



When fitting this kind of model, we make the hypothesis of homoscedasticity of the residuals.

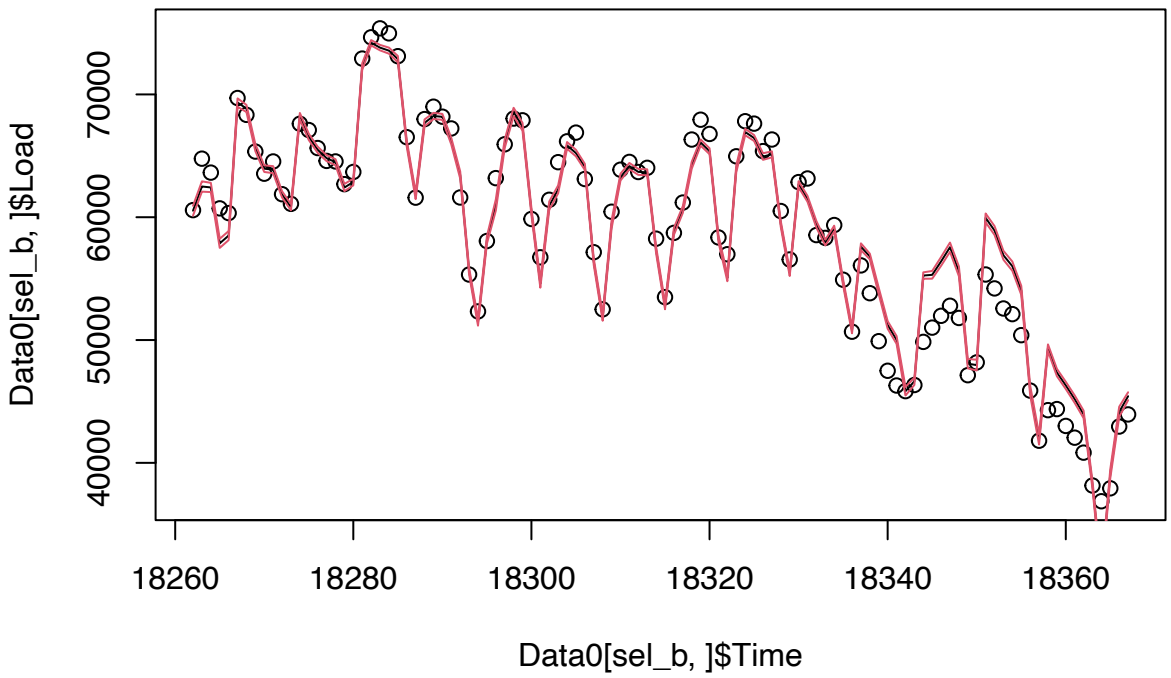
In contrast, if we plot the boxplots of the residuals grouped by temperature intervals, we observe that the variance is not equal. The heteroscedasticity forces us to readjust the GAM. In order to treat this specific case, we add a parameter allowing the learning rate to change based on the Temperature so as to avoid these variance jumps.

```
# temp_refined_qgam_model <- refined_qgam_model(dataset = Data0[sel_a,], quantile = 0.3)
#
# saveRDS(temp_refined_qgam_model, "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predictive mode

temp_refined_qgam_model <- readRDS(file = "/Users/halvardbariller/Desktop/M1 MATH&IA/S2/Predicti

#plot(temp_refined_qgam_model, scale = F, page = 1)
#check(temp_refined_qgam_model)
# check(temp_refined_qgam_model$calibr)
```

```
pred <- predict(temp_refined_qgam_model, newdata = Data0[sel_b,], se=TRUE)
plot(Data0[sel_b,]$Time, Data0[sel_b,]$Load)
lines(Data0[sel_b,]$Time, pred$fit, lwd = 1)
lines(Data0[sel_b,]$Time, pred$fit + 2*pred$se.fit, lwd = 1, col = 2)
lines(Data0[sel_b,]$Time, pred$fit - 2*pred$se.fit, lwd = 1, col = 2)
```



As we have already explained for the choice of the quantile to minimize, with the covid, the Load was sharply shrunk. Thus, even though we have tried to choose a quantile between 0.3 and 0.5, we notice that the step when the covid began is easily noticeable. Hence, we will also have to train another model to learn from the residuals.

6. Adaptative Random Forest model

Just using the same idea of the rolling window, we apply this method to construct a random forest that will be able to incorporate the Covid most important variables.

As we may observe at this plot, we suffer from an important prediction misaccuracy when the COVID appears, but using this method and including this information we are able to adapt the model and shrink the OOB error.

7. Adaptative Boosting model

a. GBM algorithms

We used Generalized Boosted Models algorithms, for which we tested several distributions assumptions to impact the loss functions computed. The GB models use CART trees as weak-learners.

Distribution: It is used to decide the loss function. As our objective is to diversify the models to aggregate them at the end with an expert aggregation, we will try multiple losses. In particular, we will try the squared error ("gaussian"), absolute loss("laplace") and the quantile loss at level $\alpha = 0.3$.

Weights: We decided to change the weights of the training examples so as to give slightly more importance to the COVID examples. To do so, we used the Governement Response Index as reference.
Number of trees: We have tried a small learning rate (shrinkage of 0.05) and we have increased the number of iterations to obtain better results. Furthermore, to decide the number of trees (which could be seen as the number of iterations), we used the OOB error.

b. XGBoost algorithms

We used this package to try to obtain better performance as well as to try a linear model as a weak-learner.

To decide the number of iterations of the algorithms, we used cross-validation and put an early-stop criterion after 20 iterations without improvement.

III. Experts' aggregation

Strongest learners' aggregation

Our first experts' aggregation submission was made using only the two strongest learners we had initially. These were the best two models we obtained for linear and generalized additive models. We tested several aggregation rule, and the results were very encouraging.

Exhaustive experts' aggregation

The second and last experts' aggregation submission was made using all of the predictors built through our research process.
Once again, we tested several aggregation rules, and the results improved once more.

Findings

To conclude, we have tried to consider this problem with as many different ways as we could, relying strongly on the belief that the only way to properly address this modeling problem was to efficiently capture the variations throughout time.
Indeed, far from simple seasonal cycles, the COVID period has strongly interfered with the electricity load, and it was thus needed to readjust previously learned model.
Our first breakthrough in terms of performance was when we started using the adaptative ARIMA correction, updating the ARIMA model as days went by. This has proven to be so efficient that almost all of our predictors are corrected with this function in this notebook.
When realising the importance online learning would have in this problem, we quickly began to wonder how to use experts' advice to improve our forecastings. Building on that, we tried different models, different loss functions, different techniques so as to explain as most variance as we could with not one but with more than ten models.
Eventually, this proved to be efficient considering the results we obtained. Moreover, our approach has been fruitful since we observe nearly a uniform repartition of the weights between the 13 experts, showing how much model diversity mattered.

Discussion

During our research process, we have considered many techniques and ideas to model the data as good as we could. Unfortunately, we did not manage to go through with each of them, and hence mention some of them for further consideration.

Regarding feature engineering, additional data was collected composed of Google's France mobility reports during COVID. This data was reformatted and missing values were filled by random forest processes. However, we were facing constraints of unidentifiable model : we had roughly 75 features for only 60 training examples. We tried to perform a LASSO regression with the *glmnet* library so has to solve the dimensional problem, however findings were not conclusive enough to appear here.

Secondly, we mentioned tuning GAM based on variable selection obtained with Random Forests. Another idea to pursue this kind of explanatory variable selections could be to select based on the order of importance when performing PCA. Moreover, the problem of computing a gam over all the variables, apart from the computation cost, is that we risk overfitting our model. Then, we could just reduce the dimension of the explanatory variables to capture the most important information and relation between the variables, and apply a gam over these main PCA dimensions.

Thirdly, for computational reasons, we did not use Kalman filter on the tuned GAM, these last being very heavy. It would interesting though to assess the relevancy of this techniques.