Modelisation predictive

Alex Pierron - Lucas Biechy

06/03/2023

Import and data processing

Import librairy

Import Data

```
preTrain <- read_delim("Data/train.csv", delim=",", show_col_types  = FALSE)
test <- read_delim("Data/test.csv", delim=",", show_col_types   = FALSE)
```

Data processing

Transform Date and WeekDays in values

```
preTrain$Time <- as.numeric(preTrain$Date)
test$Time <- as.numeric(test$Date)
preTrain$Days <- as.numeric(factor(preTrain$WeekDays))
test$Days <- as.numeric(factor(test$WeekDays))
```

```
head(preTrain)
```

Check missing values

```
sapply(preTrain, function(x) sum(is.na(x)))
```

```
sapply(test, function(x) sum(is.na(x)))
```

No missing values find, we can use this directly all the dataset.

```
summary(preTrain)
```

Creation of a validation set

As well as in courses, we have decided to create a validation set on data from January to April 2020 because this period includes a period with and without containment, which is a priori a good estimator of what will follow until January 2021. This validation set will be used to compute our generalization error estimator. However, lacking Covid data in the train set, we will redefine the final models on the preTrain (i.e. validation set and train set), with hyperparameters optimized on validation set, because several features are null in the whole train set, which creates a loss of information that we think is useful to predict the test set. We believe that this is a good compromise between optimizing the hyperparameters and training our model over a period including the Covid.

```
train <- preTrain[preTrain$Year<=2019,]
val <- preTrain[preTrain$Year>2019,]
```

Data visualisation

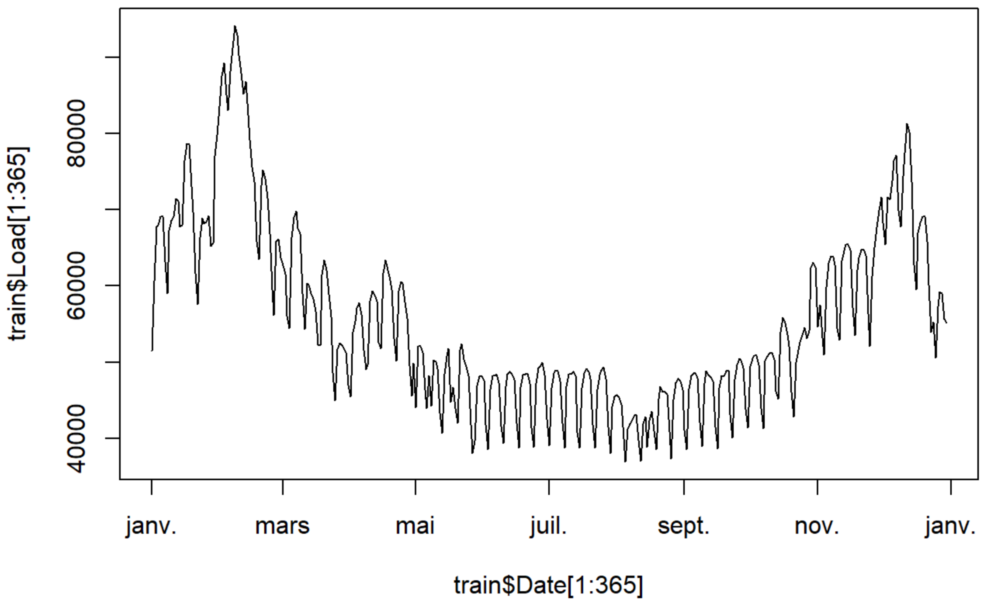
Trend

```
plot(train$Date, train$Load, type='l', xlim=range(train$Date, test$Date))
```

We can see with this plot that the electric load has no linear tendency and no tendency to be multimodal. It seems more like the consumption follows a time series model. We will therefore focus our methods to try to predict this pattern.

One year representation

```
plot(train$Date[1:365], train$Load[1:365], type='l')
```



The consumption is minimal during the summer period and maximal during the winter period. We also have a break in August, during the common vacations. Therefore the variables “summer_break” and “christmas_break” as well as “toy” and “month” will be studied.

```
plot(train$toy, train$Load, pch=16,  col=adjustcolor(col='black', alpha=0.2))
```

We can see that consumption seems fairly predictable during the summer. The three bands from 0.4 to 0.8 represent respectively the days outside the weekend, Saturday and Sunday (cf just below).

Days representation

```
boxplot(Load~WeekDays, data=train)
```

```
boxplot(Load~BH, data=train)
```

Indeed the average consumption is smaller on weekends, however it remains the same on other days. Moreover the consumption is significantly lower during the bank holidays. We will therefore study the effect of Saturday, Sunday and Bank Holidays rather than all other days.

Meteo effect

```
plot(train$Date, train$Load, type='l', ylab='')
par(new=T)
plot(train$Date, train$Temp, type='l', col='red', axes=F,xlab='',ylab='')
```

```
plot(train$Date[1:365], train$Load[1:365], type='l', ylab='')
par(new=T)
plot(train$Date[1:365], train$Temp[1:365], type='l', col='red', axes=F,xlab='',ylab='')
```

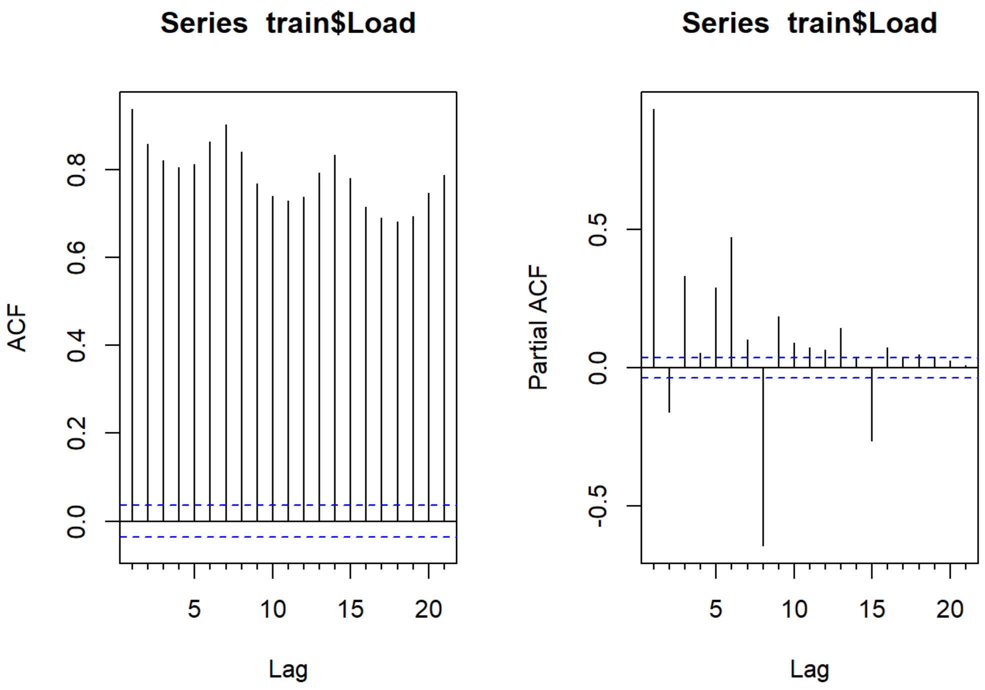
```
plot(train$Temp, train$Load, pch=3,  col=adjustcolor(col='black', alpha=0.25))
```

```
cor(train$Temp, train$Load); cor(train$Temp_s99, train$Load);cor(train$Temp_s99_max, train$Load);cor(train$Temp_s99_min, t
rain$Load); cor(train$Temp_s95, train$Load);cor(train$Temp_s95_max, train$Load);cor(train$Temp_s95_min, train$Load)
```

We see with these 3 figures and this console log that there is a strong negative correlation between the temperature (smooth and unsmooth) and the consumption, which seems linear (in two parts). However, the smooth min and max variables do not seem to increase the correlation significantly, to be tested with statistical tests.

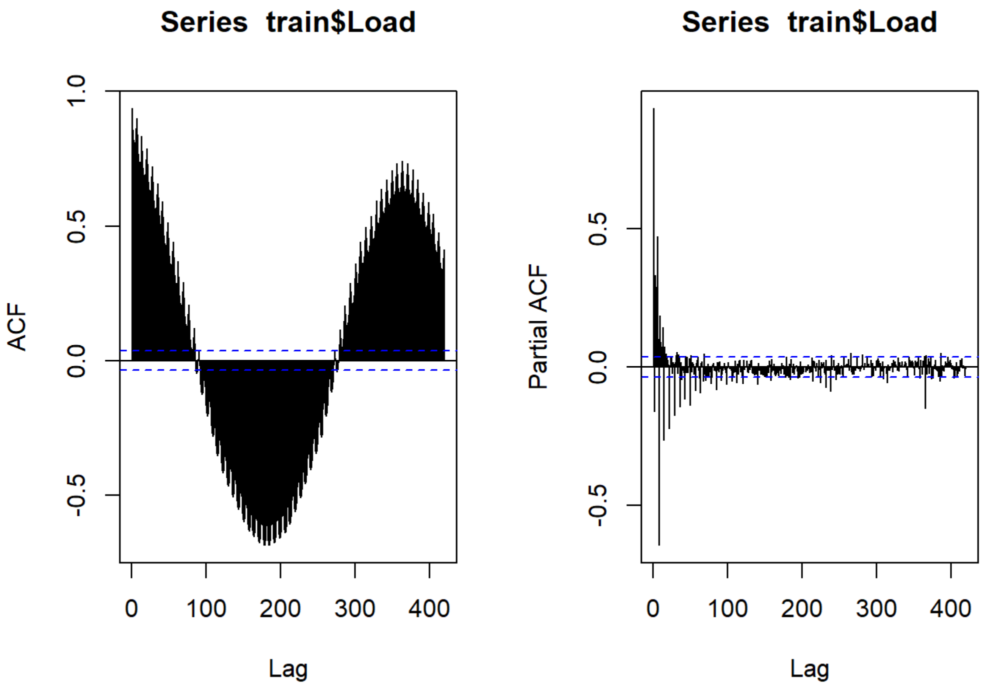
Lag

```
par(mfrow=c(1,2))
Acf(train$Load, lag.max=7*3, type = c("correlation"))
#abline(v=8, col = adjustcolor(col='grey', alpha=0.6))
Acf(train$Load, lag.max=7*3, type = c("partial"))
```



```
#abline(v=8, col = adjustcolor(col='grey', alpha=0.6))


par(mfrow=c(1,2))
Acf(train$Load, lag.max=7*60, type = c("correlation"))
#abline(v=366, col = adjustcolor(col='grey', alpha=0.6))
Acf(train$Load, lag.max=7*60, type = c("partial"))
```



```
#abline(v=366, col = adjustcolor(col='grey', alpha=0.6))
```

We can see that autocorrelation is very important across days, whether it is across the week or the year. We also notice that there are negative peaks of partial autocorrelation on the same day of the following week or year.

```
plot(train$Load.7, train$Load, pch=3)
```

```
plot(train$Load.1, train$Load, pch=3)
```

```
cor(train$Load.1, train$Load)
```

```
## [1] 0.9363499
```

```
cor(train$Load.7, train$Load)
```

```
## [1] 0.9027451
```

The correlation between Load, Load1 and Load7 is very strong and Load seems to depend linearly on Load1 and Load7. It will therefore be necessary to take into account the daily and weekly temperature inertia in our model.

First approach : Linear Model

After getting familiar with our data set and creating a train and a validation set, we first use linear models to see if they are relevant. We tried several combination of factors and we ended up with using truncatures for the temperature and a Fourier series to modelize the periodicity of the electrical consumption.

```
w<-2*pi/(365)
Nfourier<-50
Data0$Temp_trunc1 = pmax(Data0$Temp-15,0)
Data0$Temp_trunc2 = pmax(Data0$Temp-20,0)
for(i in c(1:Nfourier))
{
  assign(paste("cos", i, sep=""),cos(w*Data0$Time*i))
  assign(paste("sin", i, sep=""),sin(w*Data0$Time*i))
}

cos<-paste('cos',c(1:Nfourier),sep="",collapse=",")
sin<-paste('sin',c(1:Nfourier),sep="",collapse=",")

Data0<-eval(parse(text=paste("data.frame(Data0,",cos,",",sin,")",sep="")))

Nfourier<-30
lm.fourier<-list()
eq<-list()
for(i in c(1:Nfourier))
{
  cos<-paste(c('cos'),c(1:i),sep="")
  sin<-paste(c('sin'),c(1:i),sep="")
  fourier<-paste(c(cos,sin),collapse="+")
  eq[[i]]<-as.formula(paste("Load~ WeekDays2 + Temp + Temp_trunc1 + Temp_trunc2+",fourier,sep=""))
  lm.fourier[[i]]<-lm(eq[[i]],data=Data0)
}
equation <- eq[[15]]
equation <- buildmer::add.terms(equation, "Load.1")
equation <- buildmer::add.terms(equation, "Load.7")
```

Here we used i = 15 after having tested all possibilities on the validation set and found out that the best accuracy is obtained with a Fourier serie with 15 terms. Nevertheless this model gave us a score around 1470 when we submitted our result on Kaggle.

We quickly abandon the linear models because of the introduction of a more permissive type of model, The Generalized Additive Models or GAMs.

Generalized Additive Model (GAM)

Let y be the response variable we are trying to model and $x_1, x_2, \ldots, x_p$ the explanatory variables. The GAM model takes the following general form:

$$y = f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p) + \text{Îr}$$

where Îľ is the random error and f_i are nonlinear functions of the explanatory variables. The f_i functions are often represented as splines, which are piecewise polynomial functions.

In the GAM model, the f_i functions are estimated by conditional likelihood maximization, assuming that the error Îľ follows a specified probability distribution (e.g., a normal distribution). To estimate the f_i functions, an optimization technique called penalized spline regression (P-splines) is used, which imposes a smoothing constraint on the functions to avoid overfitting.

The GAM model is flexible and can capture non-linear relationships between the explanatory variables and the response variable. It can also be extended to include categorical explanatory variables or interactions between explanatory variables.

As seen in course, we use the GAM models with the following equations. GAMs are very useful in our case because it allows to take smooth functions of covariates as arguments. We can also use bivariate smooth effect for the interaction between two variables (with the command line s() below for both univariate and bivariate smooth effect). These smooths functions can be tuned with a specific number of splines and the type of spline used : these two characteristics are hyperparameters. Their values are found through the use of the summary once the model trained. For this specific task, we use our train set.

The strategy we adopted to find the best combination is to put as many variables and smooth effects as possible (while steel being relevant to us) and then play with the summaries to tune our model. Because of the high complexity of the problems, we prefered this downsizing approach because it was easier to visualize each step. It is important to note that we were removing or changing on parameter at a time.

The equations we introduce here represent milestones of this research. every once in a while we submitted our result on kaggle on the test set to make sure we were going the right way and not just overfitting on the train and validation data.

```
eq<-as.formula(paste("Load~ WeekDays2 + s(toy,Temp,k=12) + I(Temp^2)+
                            s(Load.1,Load.7)+Summer_break +
                            s(Temp_s95,Temp_s99) + BH +
                            Temp_s95_min + Temp_s99_min +
                            Temp_s95_max + Temp_s99_max",sep=""))
```

Score obtained : around 965

Another feature of GAMs is the possibility to group a variable using an other one. This proves to be very efficient on both train and test set. It allowed us to gain almost 60 points of score on the visible test set on Kaggle.

```
equation = Load ~ WeekDays + s(toy,Temp,k=12) +
  te(Load.1,Load.7,k=7)+ s(Load.1,by =Summer_break) +
  s(Load.1,by =Christmas_break)+ s(Load.7, by = Summer_break,bs='cr') +
  s(Temp_s95,Temp_s99) + BH + s(Load.1,by = WeekDays2) +
  Temp_s95_min  + s(Load.7, by = WeekDays2) +
  Temp_s95_max  + s(Load.1,by = Month) + s(Load.7,by = Month)
```

Score Obtained: around 904

```
equation = Load ~ WeekDays + s(toy,Temp,k=12) +
  te(Load.1,Load.7,k=7)+ s(Load.1,by =Summer_break, bs='cr') +
  s(Load.1,by =Christmas_break,bs='cr')+ s(Load.7, by = Summer_break,bs='cr') +
  s(Temp_s95,Temp_s99) + BH + s(Load.1,by = WeekDays2,bs='cr') +
  Temp_s95_min  + s(Load.7, by = WeekDays2,bs='cr') + s(Load.1,by = Month,bs='cr') + s(Load.7,by = Month,bs='cr')
```

Score obtained on Kaggle: 870

One issue we had all along this project is how to have a relevant measure for our accuracy. We often had the case where our solutions worked better on the train and valid set than on the test set (which is normal) but we also had the opposite case ! This phenomenon should be explained by the fact that the test set is only during the Covid period and therefore data and actual consumption might be very different from our train data. Our train data are mostly out of Covid period and several data collected are therefore unusable (for example GovernmentResponseIndex).

For evaluating our models, we use both our validation set and the visible test set on kaggle by submitting our results on it.

GAM with Arima

To enhance our GAM model, we use ARIMA to reduce residuals and improve our model. For this we use Auto-Arima to speed up the calibrating process and get better results. Because we do not know the real consumption on the day to predict in our test set, we have to use the residuals between the day n we are predicting and the day n-1 for which we know the exact consumption (since it is now in the past). Therefore this hypothesis exposes our model to the case of outliers or discontinuity between two days. This has to be reminded when we will measure our precision.

Score obtained on kaggle : 738

As mentioned above the score we had on Kaggle was a great improvement of performance even though it is only evaluated on 30% of the test set. Because of the Covid period and how chaotic was this period, we might have a worse score when we will measure on the whole test set because of the discontinuity evoked before.

ARIMA (AutoRegressive Integrated Moving Average) is a time series model that is used to predict future values of a time series based on its past values. It is based on three elements:

AR (AutoRegressive) terms that take into account the past values of the time series to predict future values. The AR term is defined as follows:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t$$

where y is the time series, c is a constant, p is the order of the autoregressive regression, ϕ_i are the AR coefficients, and ϵ_t is the error term.

The MA (Moving Average) terms that take into account past errors to predict future values. The MA term is defined as follows:

$$y_t = c + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

where y is the time series, c is a constant, q is the order of the moving average, θ_i are the MA coefficients, and ϵ_t is the error term.

The I (Integrated) terms that take into account past differences between the values of the time series to predict future values. The I term is defined as follows:

$$\Delta y_t = y_t - y_{t-1}$$

where Δy_t is the difference between the current value of the time series and the previous value.

Combining these three terms, we obtain the ARIMA(p,d,q) model, where p is the order of autoregression, d is the order of differentiation, and q is the order of the moving average.

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d y_t = c + (1 + \sum_{i=1}^{q} \theta_i L^i)\epsilon_t$$

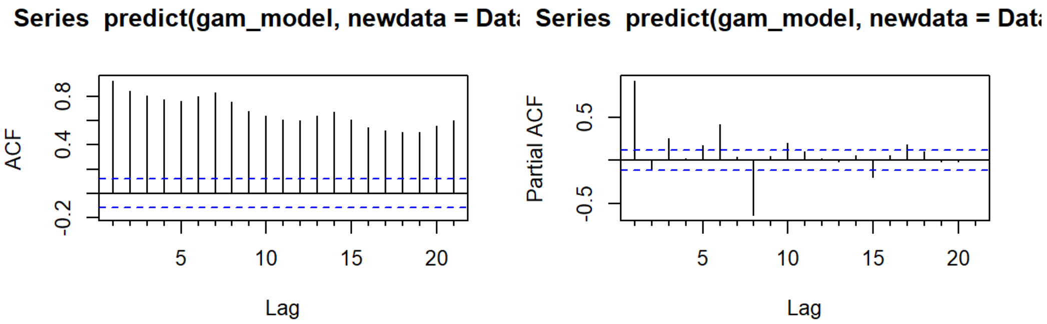
where L is the lag operator, and y_t is the time series. The ARIMA model is often used to model non-stationary time series, which require differentiation to make the series stationary.

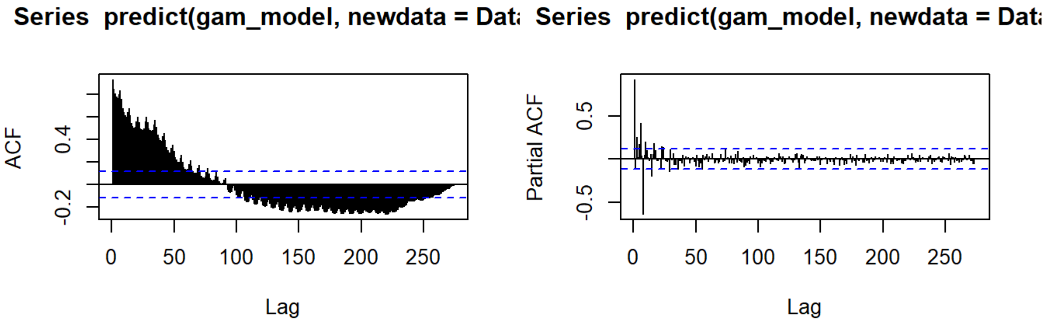
The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) are used to help identify the p and q orders of an ARIMA model. The ACF measures the linear correlation between a series and a lagged version of itself, while the PACF measures the linear correlation between two observations in the series, after removing the effect of all other observations.

ACF is used to estimate the parameter q of an ARIMA model, which corresponds to the order of the moving average. If the autocorrelation function decreases slowly and reaches zero slowly, this suggests that there is a strong correlation between each observation and observations that are a large number of periods in the past, indicating the presence of a high-order moving average.

PACF is used to estimate the parameter p of an ARIMA model, which corresponds to the order of the autoregression. If the partial correlation function decreases slowly and reaches zero slowly, this suggests that there is a strong correlation between each observation and observations that are a large number of periods in the past, indicating the presence of a high-order autoregression.

```
par(mfrow=c(2,2))
Acf(predict(gam_model,  newdata= Data1), lag.max=7*3, type = c("correlation"))
Acf(predict(gam_model,  newdata= Data1), lag.max=7*3, type = c("partial"))
Acf(predict(gam_model,  newdata= Data1), lag.max=7*60, type = c("correlation"))
Acf(predict(gam_model,  newdata= Data1), lag.max=7*60, type = c("partial"))
```





We notice that the ACF tends to 0 very slowly, so we must take a max.q as large as possible (depending on your computing power). However the PACF is very close to 0 after the 7th iteration. We should therefore set max.p 7 too.

Online GAM model with Arima

To push further this model, we tried to implement an online version of the previous algorithm. The difference is that this one will calculate a new gam model on our train set which is now the original train set plus all the day in the test set prior to the day we are predicting . We will afterward adjust an auto arima on the train set.

THis approach does not break causality because we are dealing with a time serie. We just have to make sure that the joints is correctly made and that we are using informations from the past.

Score on Kaggle : 592

This model proved to be even more effective on the visible test set on Kaggle. Despite this good news, we have to be careful because of several things : - The 30% of the test set used to calculate this score might a part of the dataset that is not that much affected by the Covid and all its implications. - For this online learning, we didn’t use Cross Validation for the residuals because it is too long to compute with our computer. Without using Cross Validation we need around 2 hours to have the final result. This time would be multiplicated by at least 3 or 4 if we use Cross Validation. It is here only a limitation due to our limited resources.

We tried to use the variable “GovernmentResponseIndex” but the results we had were not efficient on the test set part available on Kaggle. The impact of this variable can not be estimated on the train set because this set is prior to Covid.

Online Random Forest

Another model we try was an online Random Forest. We tried this model briefly because at the time we tested we already had done the online GAM-arima model. We mostly tried this model to use its results later on when we will use aggregate experts.

The first step is to construct a set of B independent decision trees. Each tree $b = 1, \ldots, B$ is constructed from a bootstrap sample of size n, obtained by drawing n observations with replacement from the training dataset.

Then, for each node in a tree, a random subset of m predictors is selected for splitting, where m is a number set by the user. The best predictor and the corresponding split point are chosen among these m predictors so as to minimize an impurity measure such as the Gini index or the Shannon entropy.

The final prediction is obtained by averaging the predictions of all the constructed trees:

$$\hat{f}\,RF(x) = \frac{1}{B} \sum b = 1^B \hat{f}_{\,b}(x)$$

where $\hat{f}_{\,b}(x)$ is the prediction of tree b for observation x.

Although a GAM model may be more accurate than a random forest model for a particular data set, it may be advantageous to use a random forest model for two reasons. Random forest models are generally more robust than GAM models because they are less sensitive to assumption violations. And random forest models are relatively easy to interpret because they are based on decision trees that are linear models.

Score on Kaggle : 1738

This model has not a great precision but since we did not try to fine-tune it, we can not give further analysis. Like we said earlier, we mostly use it later for experts aggregate.

Boosting

We also tried a boosting approach for the problem. We used the gbm package to do this. Like the Online Random Forest we tried this model after we achieved our Online GAM Arima model.

The boosting method is an ensemble method based on learning a larger number of weak learner, for instance using CART (Classification and Regression Tree) trees like the package we used (package gbm).

GBM (Generalized Boosting Models) repeatedly fit many decision trees to improve the accuracy of the model. For each new tree in the model, a random subset of all the data is selected using the boosting method. For each new tree in the model the input data are weighted in a way that data that was poorly modelled by the previous trees has a higher probability of being selected in the new tree. This means that after the first tree is fitted the model will take into account the error in the prediction of that tree to fit the next tree, and so on. By taking into account the fit of previous trees that are built, the model continuously tries to improve its accuracy.

Generalized Boosting Models have two important parameters that need to be specified by the user.

1°/ : Interaction depth : this controls the number of splits in each tree. A value of 1 results in trees with only 1 split, and means that the model does not take into account interactions between environmental variables. A value of 2 results in two splits, etc. By default the value is set to one and we do not change this value.

2°/: Shrinkage : this determines the contribution of each tree to the growing model. As small shrinkage value results in many trees to be built.

This package is furthermore using Gradient Boosting method as seen during class.

Score on Kaggle : around 1550.

We fine-tuned it a little but the results were not as good as the GAM models but improved our previous result with the online forest. This method is not used online and it might be a good idea to test to go further.

We will keep and use the results of this model for the online aggregation of experts later.

Final Model : Online aggregation of experts

We obtained our final model by using all the previous one. Each one of this model and its forecast will be named “experts”.

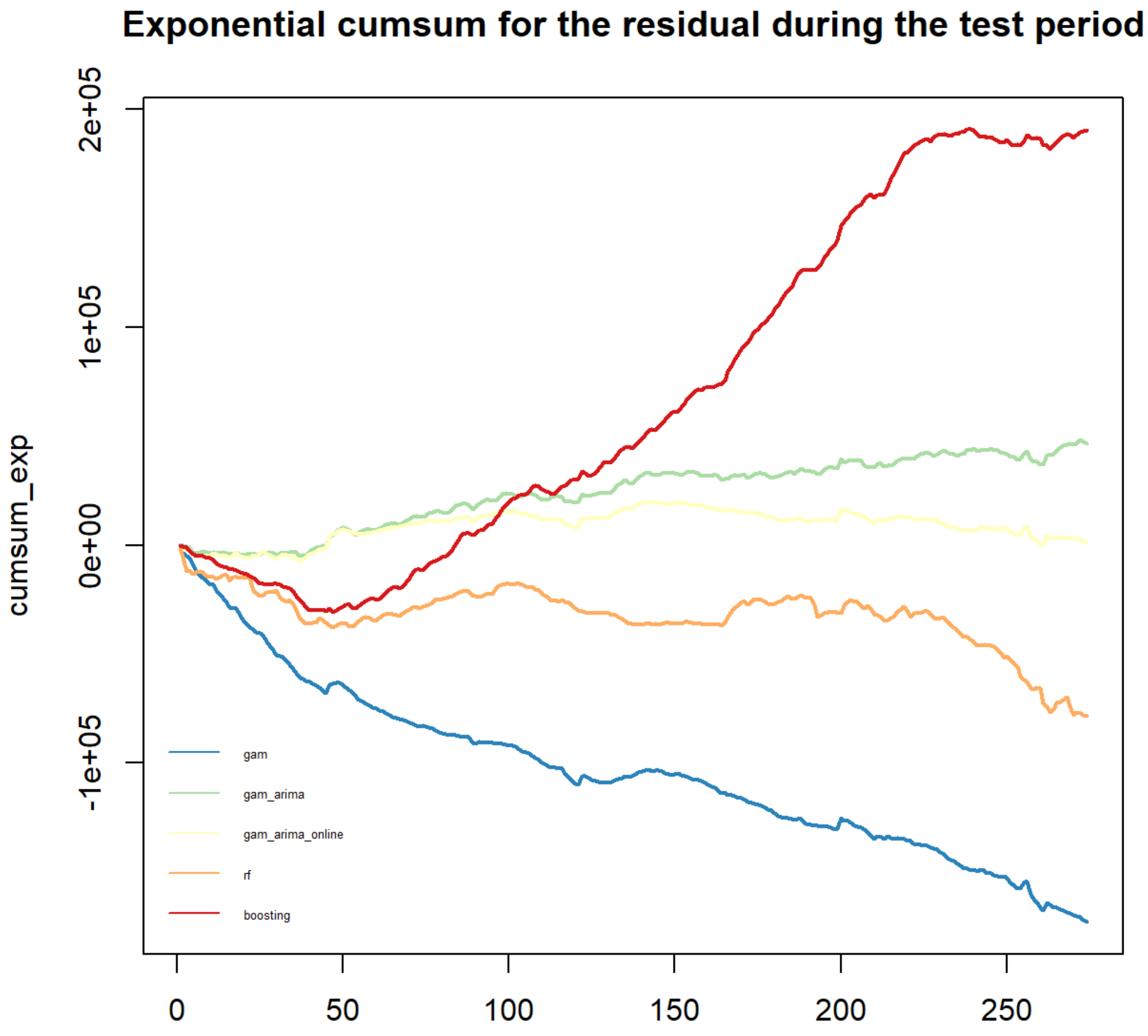
The philosophy behind the aggregation of experts is that we have access to several different forecasts for each day coming from our experts. We also assume that we have the actual targets for all the days prior to the one we want to predict.

For a day n, We are looking for producing the best aggregation possible of our experts based on the labels and forecast available for the n-1 days before. The quality of the result depends on the chosen loss function.

As seen during in course, the result of this aggregation can not be worse than the best expert we have. To have the estimation about boundaries for this method, we use the mixture function from the package opera that allow us to test several type of model. This function performs directly an online aggregation of expert with the given one.

For the final day, we use the prediction from our best individual model (which is the online GAM Arima model).

We represent the plot of the different error cumsum for each model:



We notice that the two GAM models using ARIMA correction for residuals handle them better than the other models. By seeing this, it is tempting to add a biais term to have additionnial experts that might be relevant.

```
## Aggregation rule: MLpol
## Loss function:  squareloss
## Gradient trick:  FALSE
## Coefficients:
##  gam gamarima gamarimaonline rf boosting
##    0        0              1  0        0
##
## Number of experts:  5
## Number of observations:  274
## Dimension of the data:  1
##
##         rmse   mape
## MLpol    980 0.0141
## Uniform  841 0.0131
## Aggregation rule: MLpol
## Loss function:  squareloss
## Gradient trick:  TRUE
## Coefficients:
##    gam gamarima gamarimaonline     rf boosting
##  0.201    0.235          0.212 0.0995    0.252
##
## Number of experts:  5
## Number of observations:  274
## Dimension of the data:  1
##
##          rmse   mape
## MLpol    834 0.0122
## Uniform  841 0.0131
```

```
## Aggregation rule: BOA
## Loss function:  squareloss
## Gradient trick:  TRUE
## Coefficients:
##    gam gamarima gamarimaonline    rf boosting
##  0.179    0.291          0.257 0.146    0.127
##
## Number of experts:  5
## Number of observations:  274
## Dimension of the data:  1
##
##          rmse   mape
## BOA      849 0.0126
## Uniform  841 0.0131
```

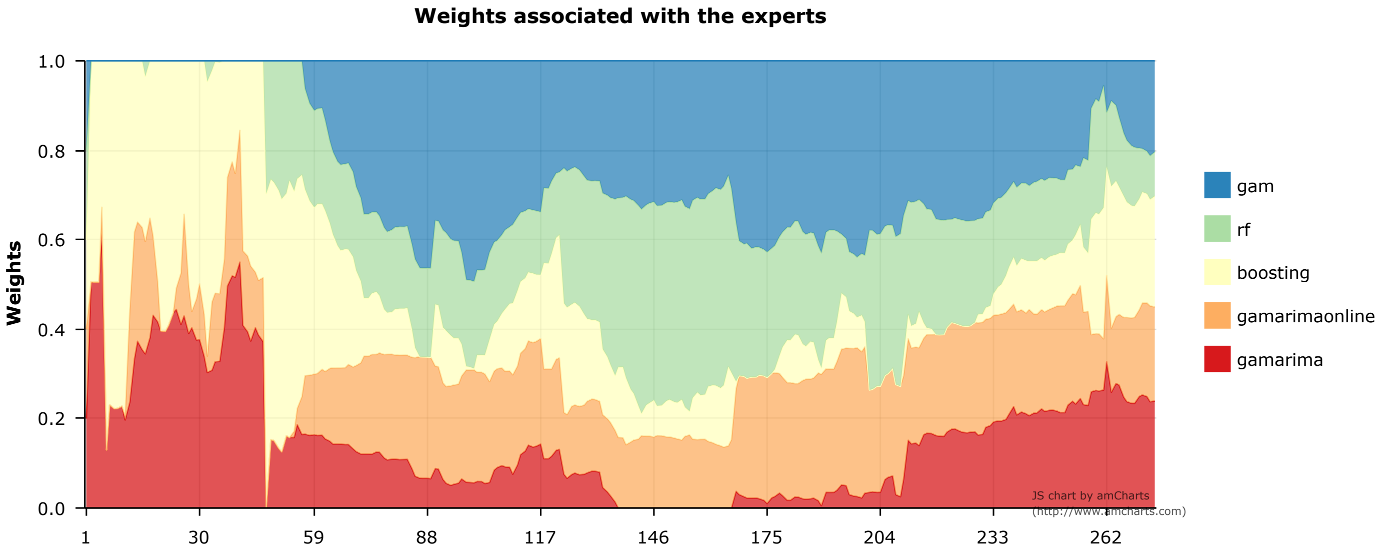
We compare the three model tested above and their summary for the aggregation of experts. We then determine that the best model is the one using the MLpol (an aggregation rule with multiple learning rates that are theoretically calibrated, Gaillard et al. 2014). The BOA stands for Bernstein online Aggregation (Wintenberger 2017). It is using learning rates that are automatically calibrated.

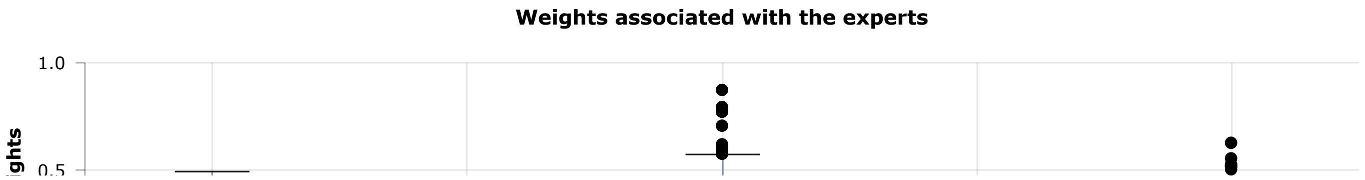
We only tested this two methods out of the large choice available.

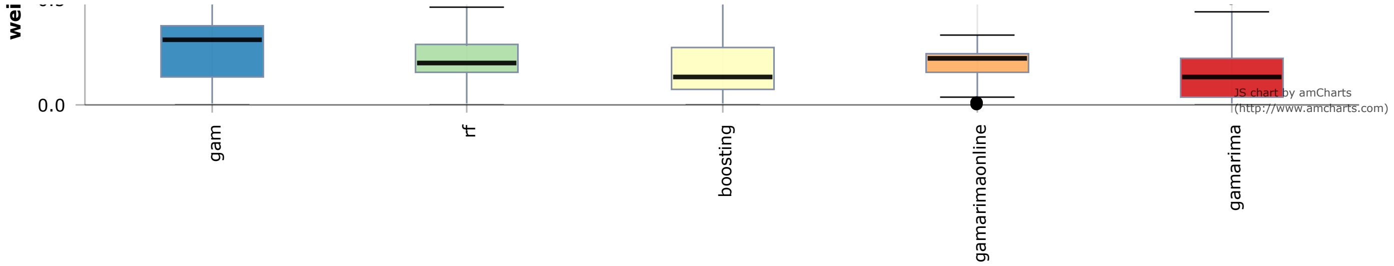
We have the following figure for how the experts are used during the online process.

```
plot(agg_mlpol_lgt)
```

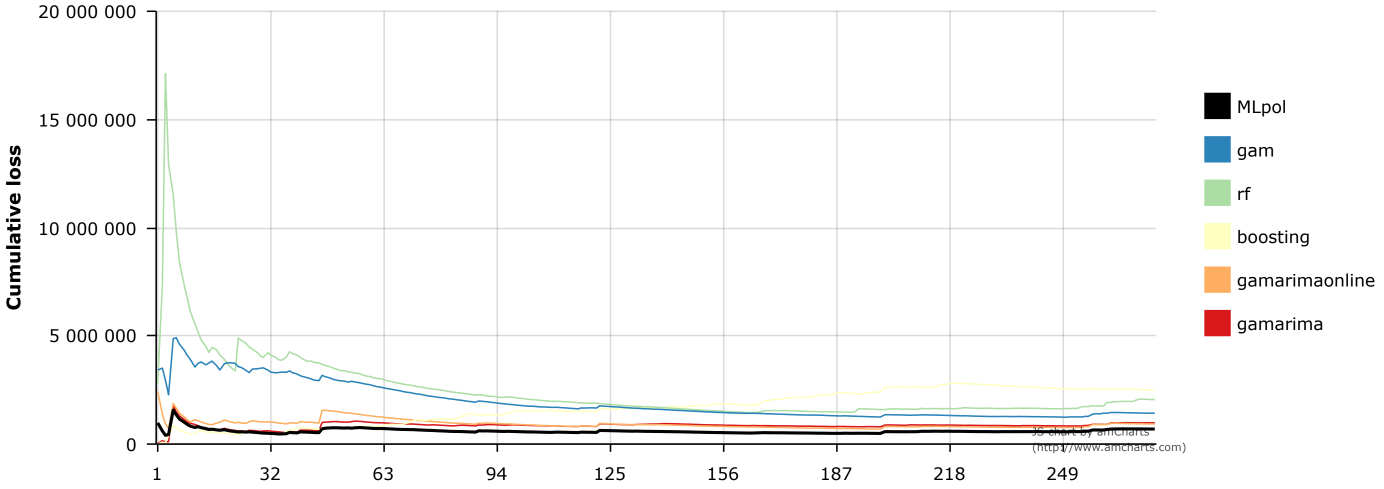
```
## Warning in par(def.par, new = FALSE): argument 1 does not name a graphical
## parameter
```



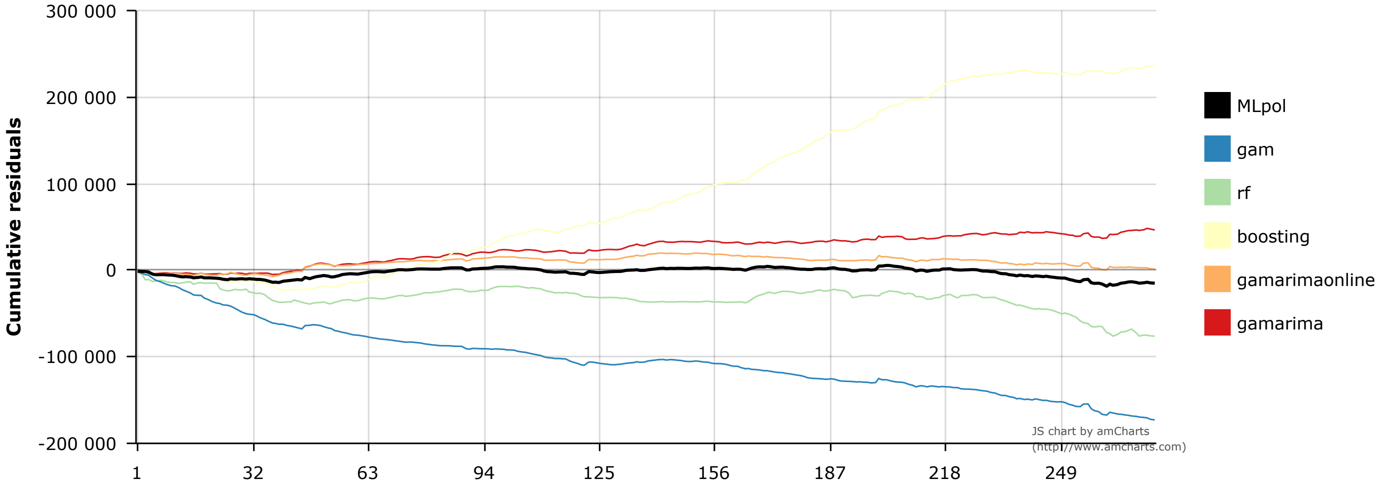




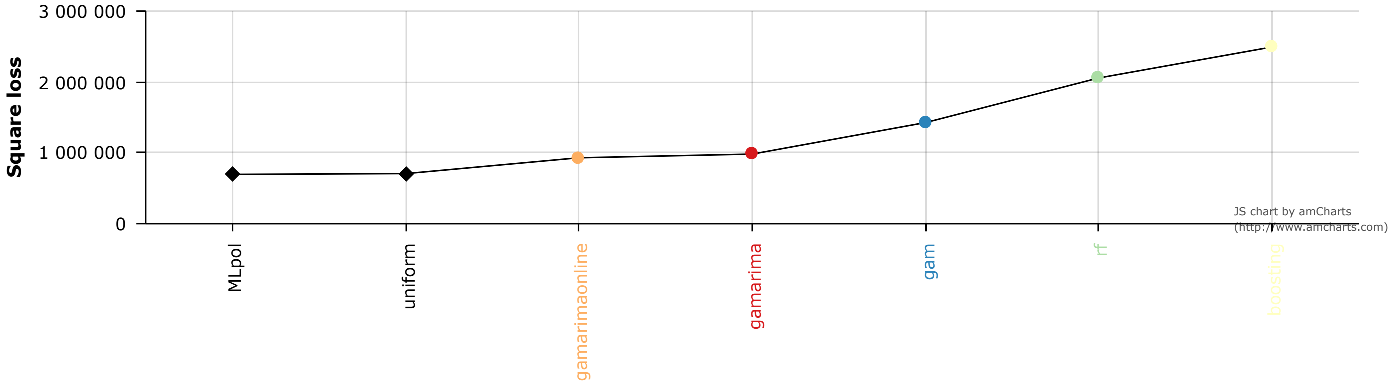
Dynamic average loss



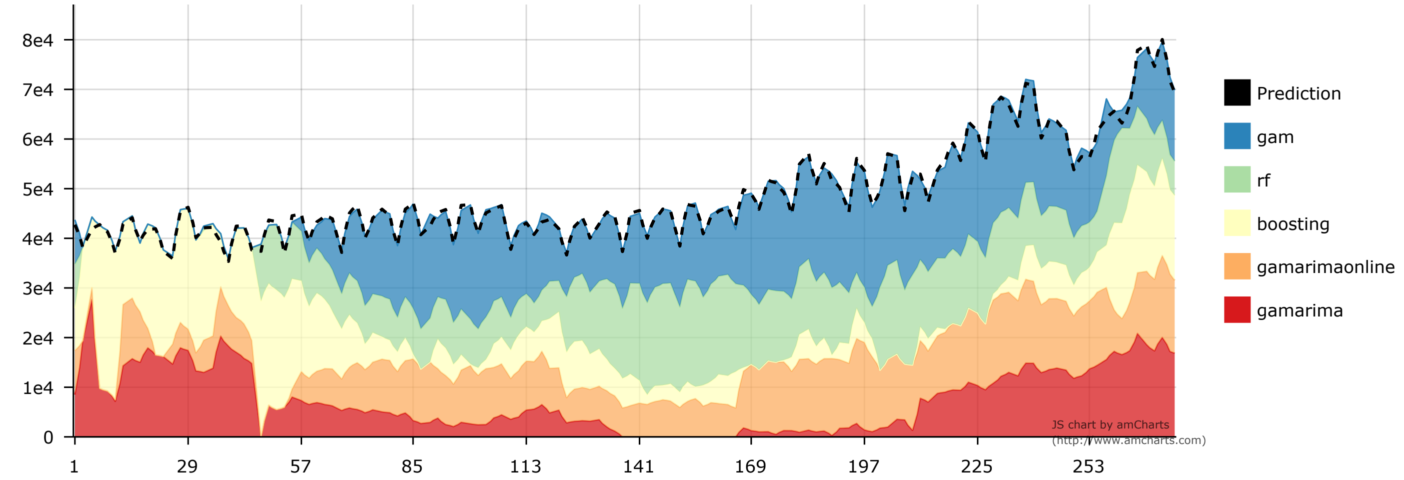
Cumulative residuals



Average loss suffered by the experts



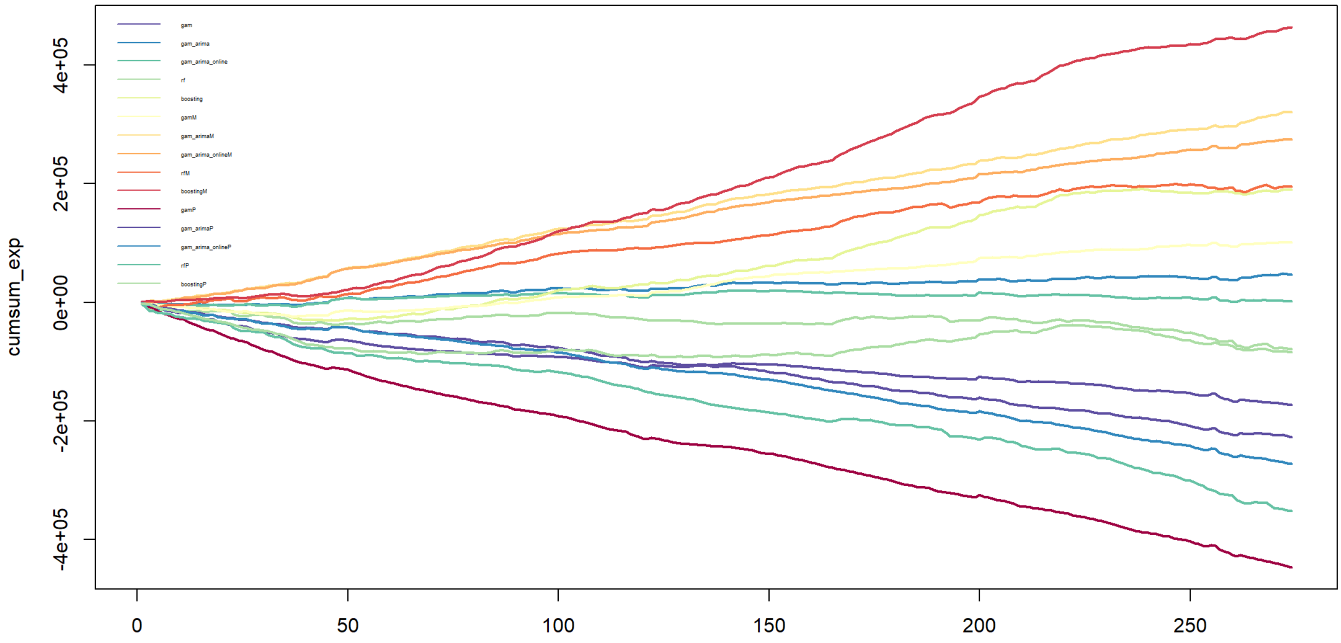
Contribution of each expert to the prediction



If we compare the cumulative residuals, we see that the MLpol model (our expert aggregation model) is globally better than all of the other even if at the end of the test set we see that our new model is slightly worse than the model with an online GAM and Arima.

To enhance our expert aggregation, we decide to create new experts by adding or substracting a biais we fixed by testing several values. Then we do the same procedure as before.

When we look at the cumulative exponential residuals, we have then:



Cumulative exponential residuals

We clearly see that some of the biaised models are now relevant and might provide extra information if we include them inside our aggregation of experts.

By doing the same thing as before, we have:

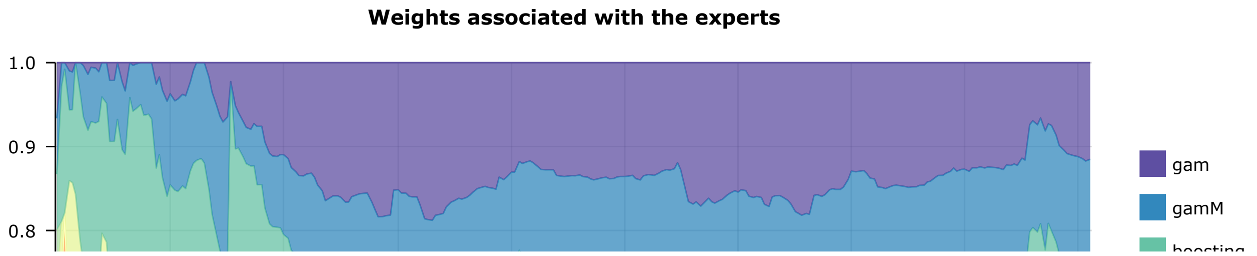
```
## Aggregation rule: MLpol
## Loss function:  squareloss
## Gradient trick:  FALSE
## Coefficients:
##  gam gam_arima gam_arima_online rf boosting gamM gam_arimaM gam_arima_onlineM
##    0         0                1  0        0    0          0                 0
##  rfM boostingM gamP gam_arimaP gam_arima_onlineP rfP boostingP
##    0         0    0          0                 0   0         0
## 
## Number of experts:  15
## Number of observations:  274
## Dimension of the data:  1
## 
##         rmse   mape
## MLpol    980 0.0142
## Uniform  841 0.0131
```

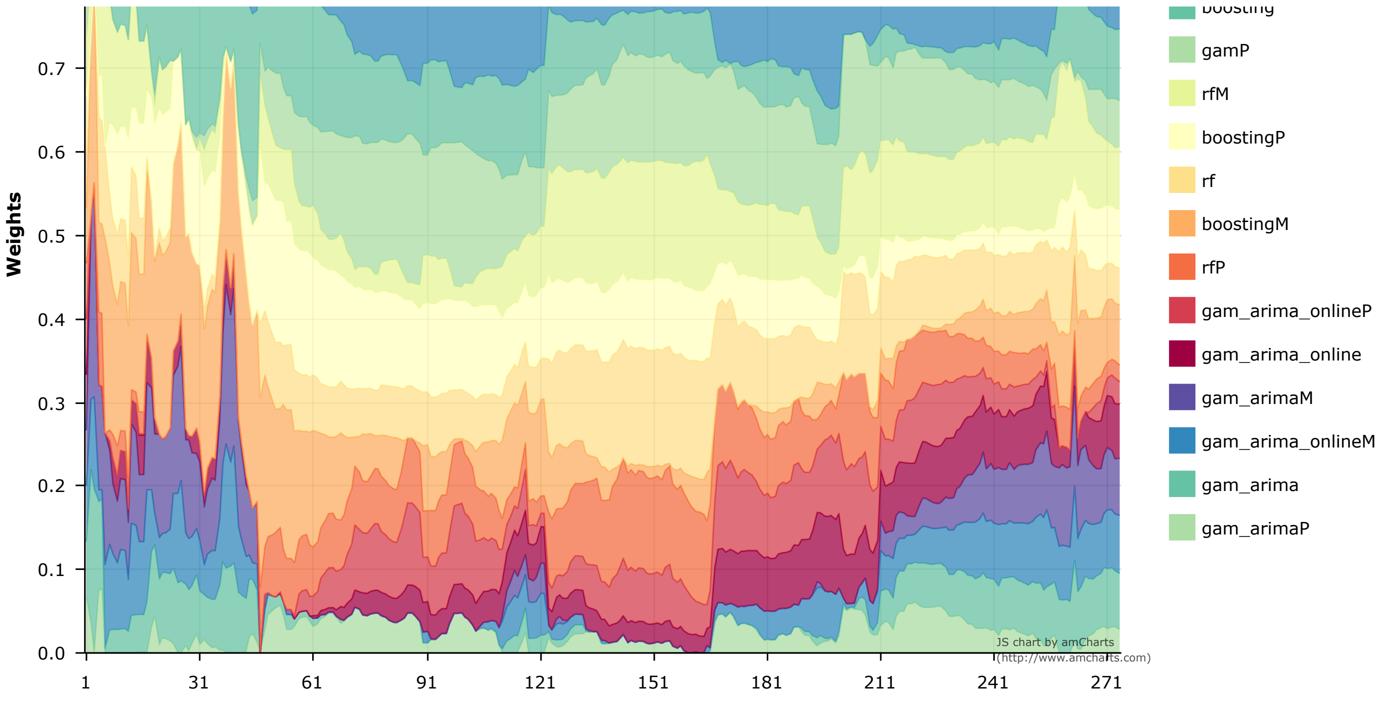
```
## Aggregation rule: MLpol
## Loss function:  squareloss
## Gradient trick:  TRUE
## Coefficients:
##    gam gam_arima gam_arima_online     rf boosting  gamM gam_arimaM
##  0.115    0.0665           0.0651 0.0441   0.0866 0.138      0.069
##  gam_arima_onlineM    rfM boostingM   gamP gam_arimaP gam_arima_onlineP    rfP
##             0.0707 0.0761    0.0738 0.0543     0.0258            0.0254 0.0195
##  boostingP
##     0.0699
## 
## Number of experts:  15
## Number of observations:  274
## Dimension of the data:  1
## 
##         rmse   mape
## MLpol    818 0.0120
## Uniform  841 0.0131
```

```
## Aggregation rule: BOA
## Loss function:  squareloss
## Gradient trick:  TRUE
## Coefficients:
##    gam gam_arima gam_arima_online    rf boosting  gamM gam_arimaM
##  0.149    0.0754           0.0862 0.043   0.0849 0.125     0.0436
##  gam_arima_onlineM   rfM boostingM   gamP gam_arimaP gam_arima_onlineP    rfP
##             0.0611 0.108    0.0315 0.0249     0.0559            0.0226 0.0235
##  boostingP
##     0.0651
## 
## Number of experts:  15
## Number of observations:  274
## Dimension of the data:  1
## 
##         rmse   mape
## BOA      838 0.0125
## Uniform  841 0.0131
```

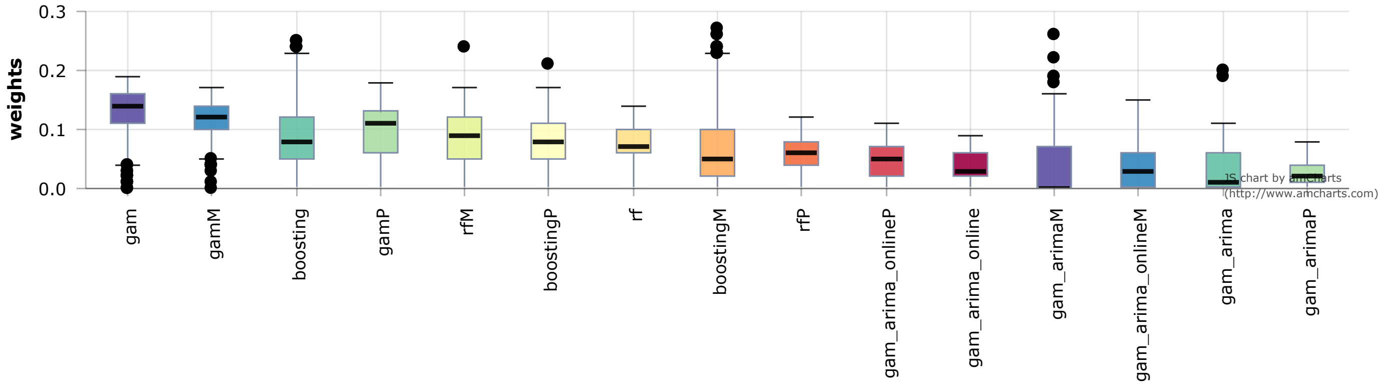
```
plot(agg_mlpol_lgt)
```

```
## Warning in par(def.par, new = FALSE): argument 1 does not name a graphical
## parameter
```

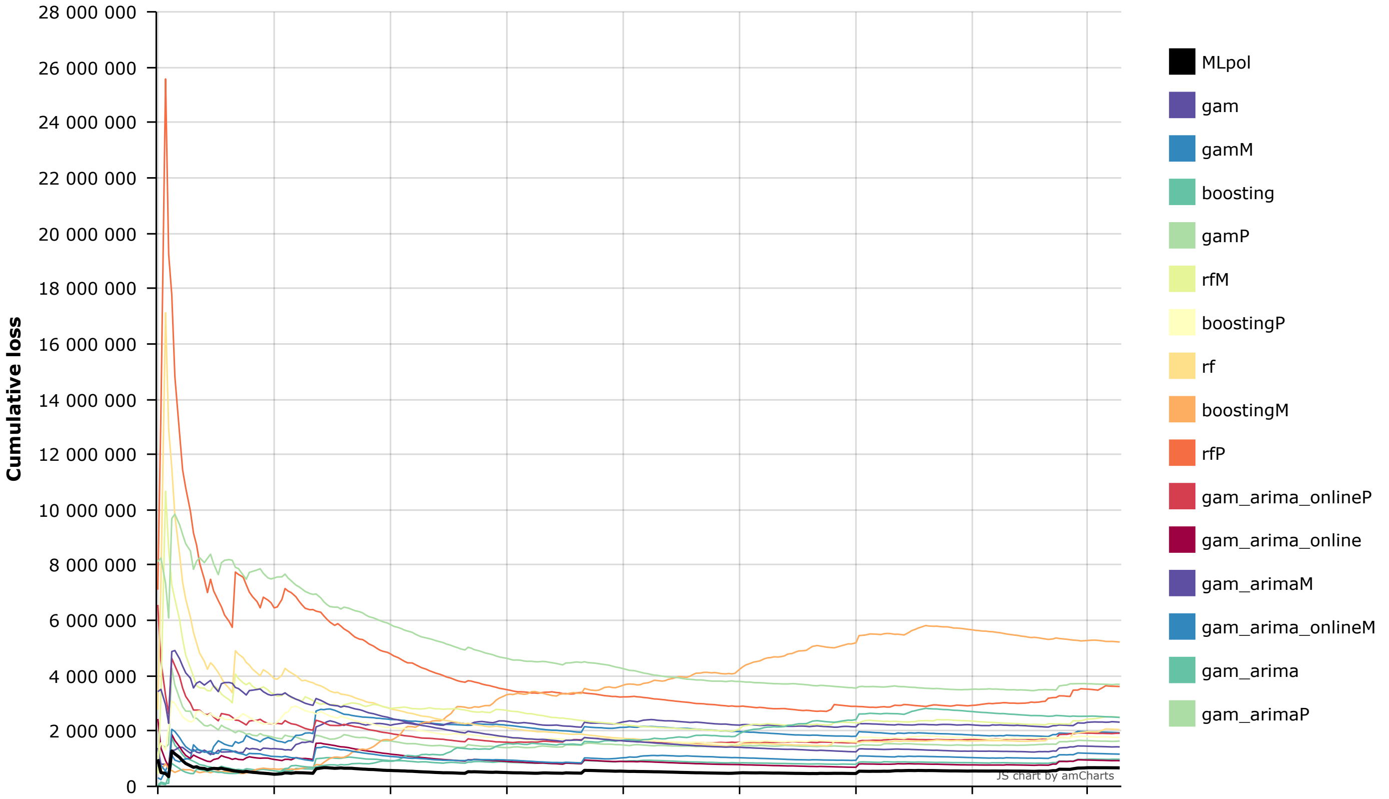




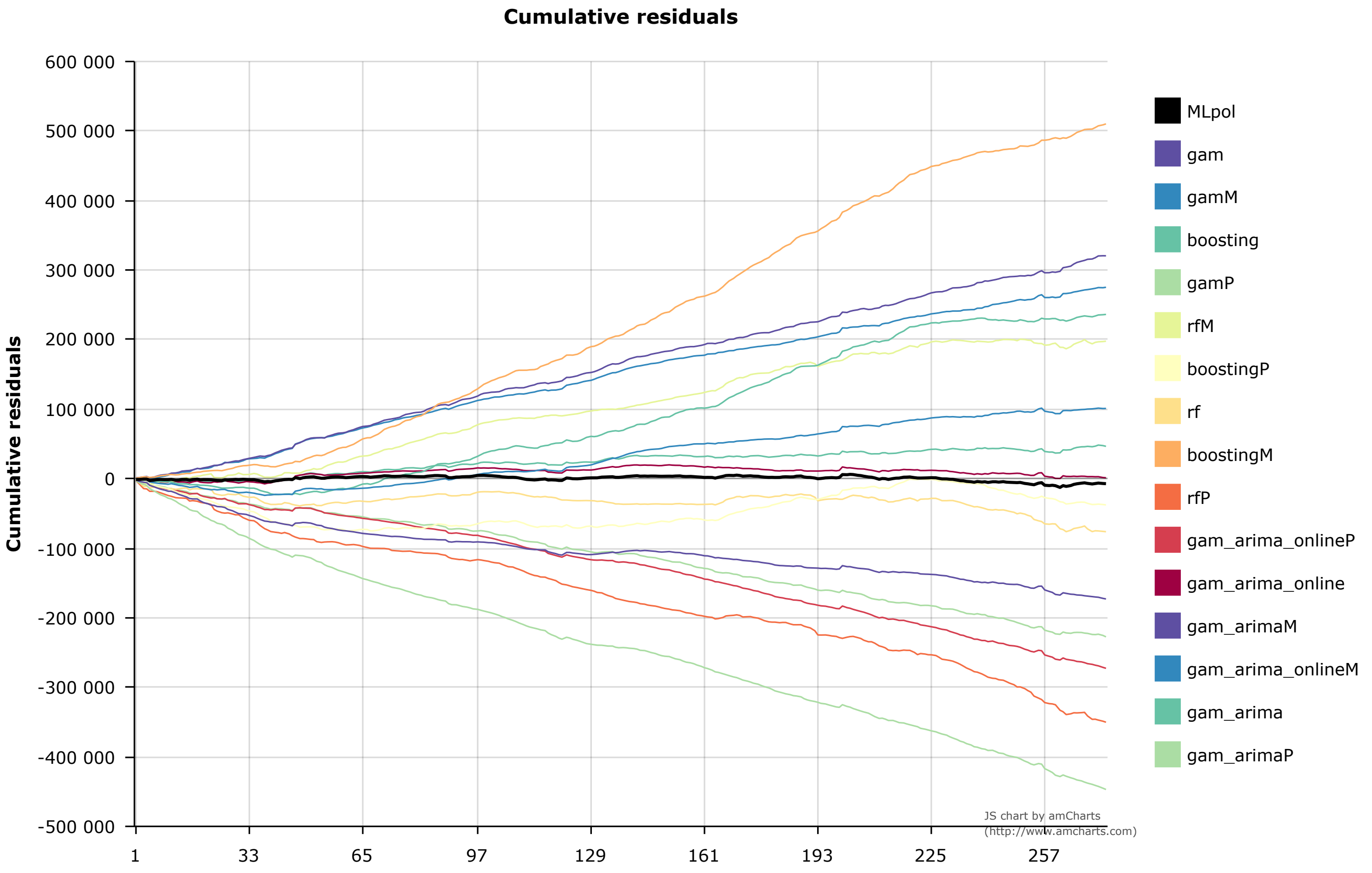
Weights associated with the experts

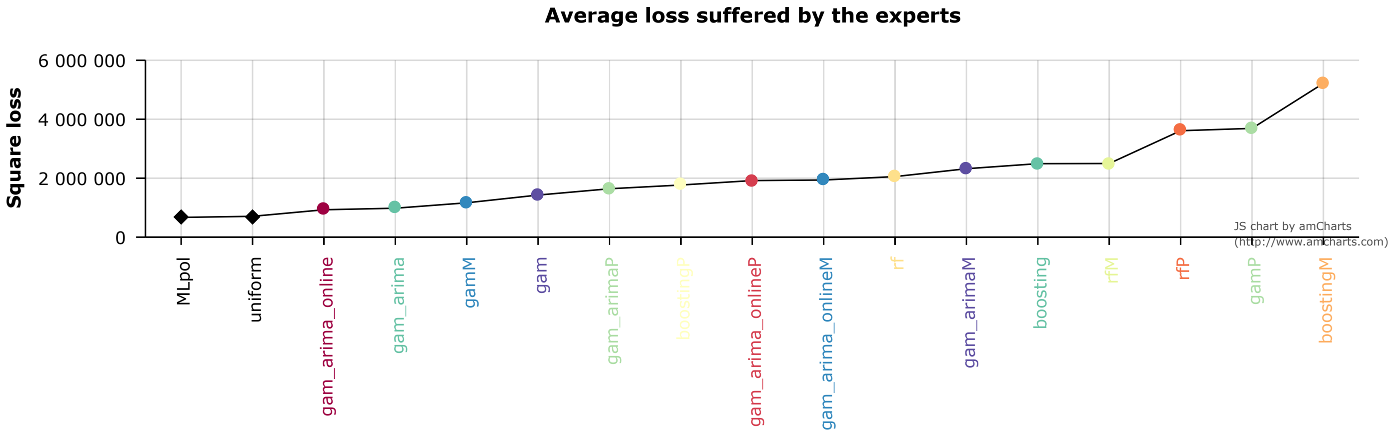


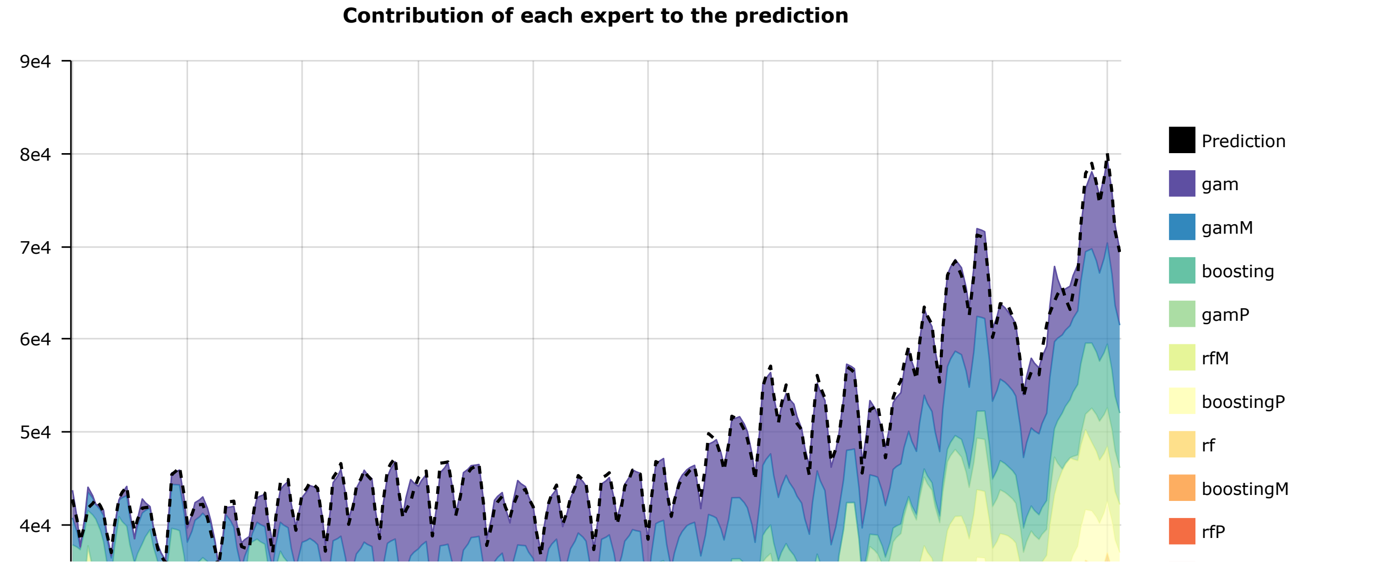
Dynamic average loss

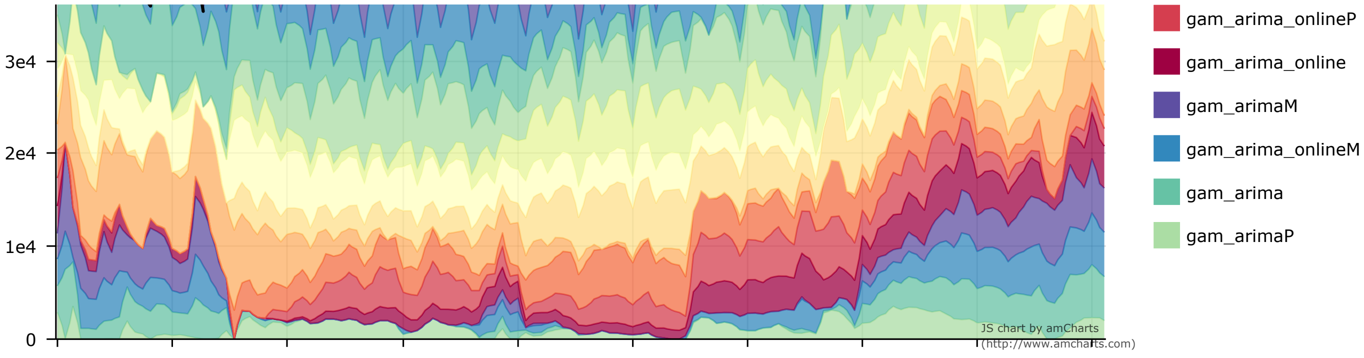


1
34
67
100
133
166
199
232
265
(http://www.amcharts.com)


Cumulative residuals
Cumulative residuals
600 000
500 000
400 000
300 000
200 000
100 000
0
-100 000
-200 000
-300 000
-400 000
-500 000
1
33
65
97
129
161
193
225
257
JS chart by amCharts
(http://www.amcharts.com)
MLpol
gam
gamM
boosting
gamP
rfM
boostingP
rf
boostingM
rfP
gam_arima_onlineP
gam_arima_online
gam_arimaM
gam_arima_onlineM
gam_arima
gam_arimaP


Average loss suffered by the experts
Square loss
6 000 000
4 000 000
2 000 000
0
JS chart by amCharts
(http://www.amcharts.com)
MLpol
uniform
gam_arima_online
gam_arima
gamM
gam
gam_arimaP
boostingP
gam_arima_onlineP
gam_arima_onlineM
rf
gam_arimaM
boosting
rfM
rfP
gamP
boostingM


Contribution of each expert to the prediction
9e4
8e4
7e4
6e4
5e4
4e4
Prediction
gam
gamM
boosting
gamP
rfM
boostingP
rf
boostingM
rfP



If we look to the residuals, we can clearly see an improvement for our final model here in black in the figures below.

Residuals from our final model are closer and better adjusted to the reality. The gap we saw at the end of the set with the online GAM Arima model is now not significative and is even “removed”.

It is confirmed by plotting the average loss for all the experts and the final model. We see that our model is better than any experts it uses which is a statement we claimed earlier.

Final Comments

To conclude this study, even if we reached a decent level of accuracy for our prediction, here are some possible way to improve our model:

We could use Cross Validation during the online GAM- Arima model.

We could furthermore fine-tune the equation used for all the GAM models.

We mostly used the raw data given to us without using new ones outside of the dataset. Some additionnal and relevant data could be included and bring further informations, allowing the model to be more accurate.