AXA Challenge
Team casaju

Juliette Achdou
juliette.achddou@telecom-paristech.fr

Salma El Alaoui Talibi
salma.el-alaoui-talibi@polytechnique.edu

Camille Jandot
camille.jandot@telecom-paristech.fr

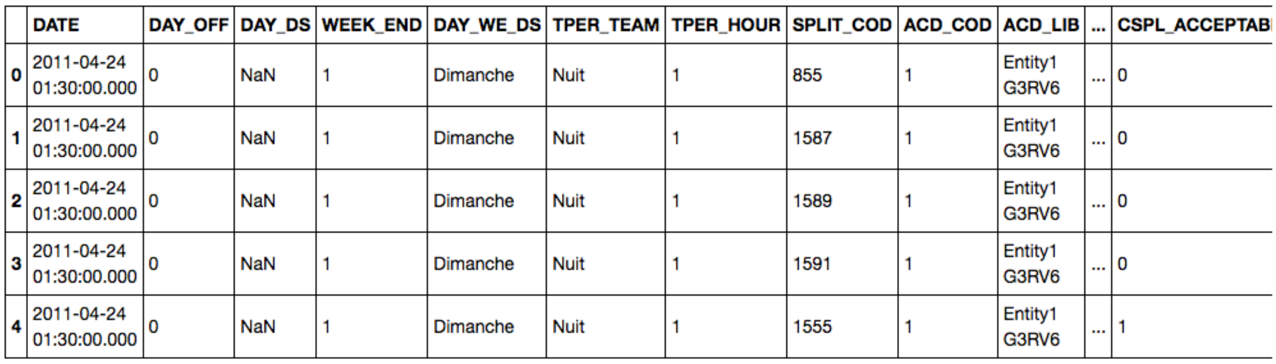
January 11, 2017

Abstract

The problem we tackled in this challenge consisted in forecasting numbers of calls received by units of a call-center, during a week, given past data on such calls.
To complete this task , after having analyzed the data and pre-processed it in a way that we could apply usual machine learning methods, we applied "natural/hand-made" methods, and ensemble methods, such as Random Forests and Gradient Boosting.

1 Introduction

The training dataset we had at our disposal is composed of $10,878,470$ rows of data. This data corresponds to Telephony Information for 2011, 2012, and 2013. Twelve weeks are missing from this data, one every four weeks, starting at the end of 2012: they are exactly the weeks we have to forecast the number of calls on.
Each piece of data is composed of 57 features: some of them related to the dates, some of them related to the nature of the calls.



	DATE	DAY_OFF	DAY_DS	WEEK_END	DAY_WE_DS	TPER_TEAM	TPER_HOUR	SPLIT_COD	ACD_COD	ACD_LIB	...	CSPL_ACCEPTAB
0	2011-04-24 01:30:00.000	0	NaN	1	Dimanche	Nuit	1	855	1	Entity1 G3RV6	...	0
1	2011-04-24 01:30:00.000	0	NaN	1	Dimanche	Nuit	1	1587	1	Entity1 G3RV6	...	0
2	2011-04-24 01:30:00.000	0	NaN	1	Dimanche	Nuit	1	1589	1	Entity1 G3RV6	...	0
3	2011-04-24 01:30:00.000	0	NaN	1	Dimanche	Nuit	1	1591	1	Entity1 G3RV6	...	0
4	2011-04-24 01:30:00.000	0	NaN	1	Dimanche	Nuit	1	1555	1	Entity1 G3RV6	...	1

Figure 1: Columns of initial dataframe

Our task is to give a prediction of CSPL_CALLS, received calls, for every 30-minutes time slots for the missing weeks, training only on past data, without looking at the future data, as our model needs to be deployable in a real life situation.
The evaluation is made using the LinEx Loss which penalizes underestimation more than overestimation.

2 Data Analysis

2.1 Seasonality

To verify our intuition towards seasonality, we proceeded to an ARIMA decomposition.

Using these kinds of graphs, we could see that the data presented a strong seasonality in terms of weeks. The ARIMA model decomposes a time-series into a seasonality signal: a periodic signal, a trend: a signal represent the global evolution of the signal using a moving average, and a residual. The trend is also not to be underestimated, as we see there can be a strong increase over the years, on a precise assignment.
Here, there is a strong residual, which is not a good sign for ARIMA models, but we have to keep in mind that there are some missing values that affect the result.

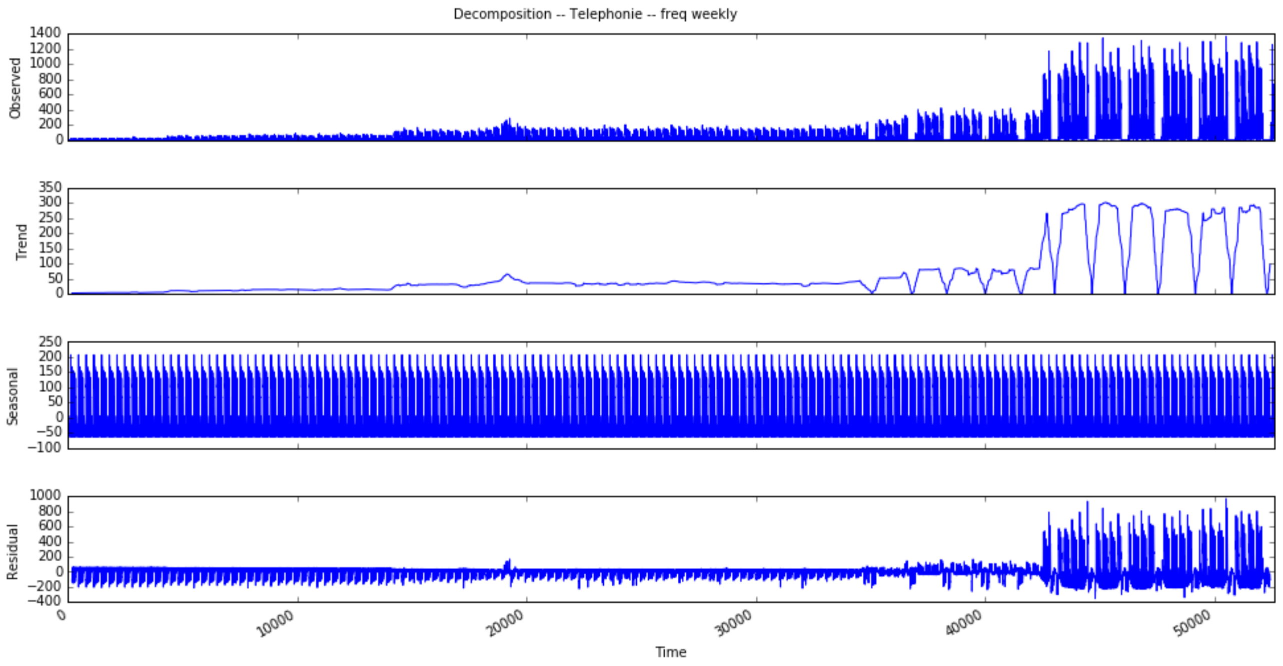


Figure 2: Seasonal Decomposition with ARIMA

We can also check the global weekly seasonality by plotting the sum of the calls par day, per assignment on a month (Figure 3).

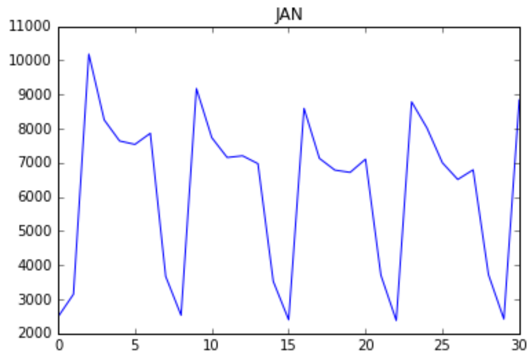


Figure 3: Total Number of calls per day in January 2011

Here, biggest peaks are on Mondays, then it drops slowly until a second small peak on Fridays, and it drops drastically on week-ends.

Of course not all assignments show this level of structure/seasonality, but it is also due to assignments where very few calls happen: assignments which do not reach 10 calls a day tend to seem more random.

Actually, intuitively, there is a multiple seasonality: yearly, weekly, daily.
We can see the usual evolution of calls for assignment "Telephonie" in (Figure 4). Most of the calls are made between 9 AM and 11 PM. There is a peak of activity around 11 AM.
Of course, not every assignment has such a structure, as some assignment concentrate their activity at night for example...

2.2 Holidays

By watching the holidays more closely, we see that the very low activity on a holiday reverberates on the next day. Thus, processing holidays separately seems important.

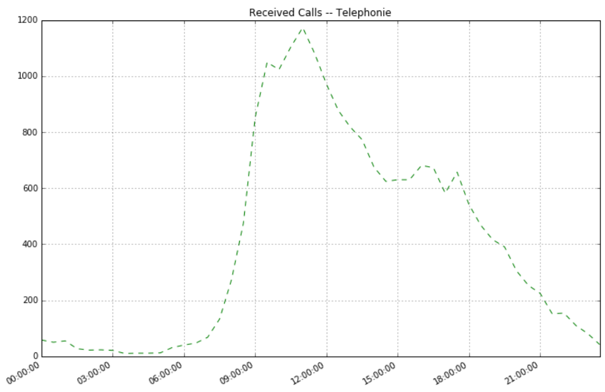


Figure 4: Daily Seasonality

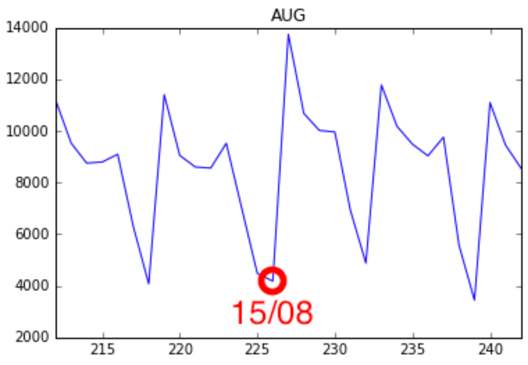


Figure 5: Total number of calls per day in August 2011 (showing a holiday)

2.3 Other Insights

The assignments which show the largest number of calls per day are “Téléphonie”, “Tech AXA”,“Tech Inter” “Services”, we can guess that they are the ones that we will have the largest errors on.
There are unpredictable assignments, like “Evènements”, which we do not have to predict,
There is also a huge peak on “Téléphonie”, which will probably be hard to predict: see (Figure 6).

By looking at the different assignments , we can see that some of the assignments start after January 2011, or stop before 2013. “Gestion - Renault” is always zero, “Gestion ” seems to have started in September 2012, “Gestion - Amex” seems to stop in August 2012.

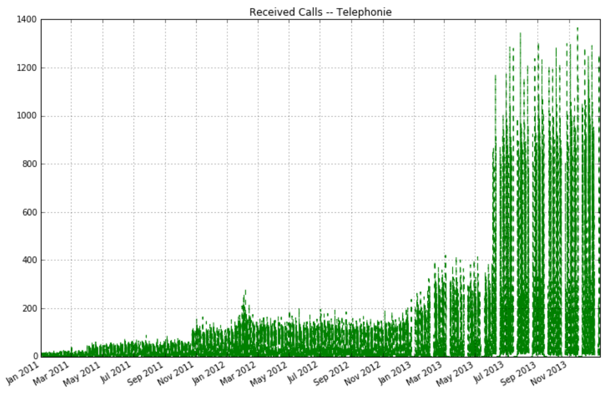


Figure 6: Peak in Téléphonie data during summer 2013

3 Preprocessing / Data Cleaning

We have observed that a lot of features contain only one value, or no value at all: "ACD_COD", "ACD_LIB", "ACD_INCOMPLETE", for example.
These features will be dropped.
We also have observed that there are missing data (all (date, assignment) pairs are not covered), and some (date,assignment) pairs are represented twice. We chose to replace missing data by zero, as it seemed to create better models, and we assumed that people did not fill the number of received calls when there were none. We chose to sum the occurrences of duplicated couples (date,assignment).
In the end, we dropped all features we could not obtain for the submission file.

4 Simple Models

After analyzing our data, we chose to build simple models, that do not involve any machine learning techniques to see whether it was possible to predict the number of received calls to predict using only data from the close past. The models described in this section are implemented in the file `dummy_prediction.py`.

4.1 Predicting the number of calls at previous week

Looking at the curves of evolution of number of calls per assignment, we noticed that the weekly pattern was really noticeable. That is why we first attempted to do a very simple prediction: forecasting the number of calls at the same time slot of the previous week, for the same assignment. Doing so, our error on the leaderboard was $1,215$. Overestimating our prediction by multiplying by 2, we reached an error of 2.59.

A drawback of this method is that it is very sensible to irregularities in the training set: if one week, in a given assignment, at a given time slot, there is a sudden peak or a sudden gap, we will predict the same unusual pattern for the newt week. Moreover, it does not take into account the evolution of the volume of calls. Next model aims at correcting these loopholes.

4.2 Using more robust but still simple models

We tried several methods to make the model more robust to peaks and gaps, still keeping the model very simple. For example, instead of taking the value of the number of calls at week W-1, we tried to take the maximum of the numbers of calls for the couple (time slot, assignment) at W-1, W-2 and W-3. The *maximum* metric was chosen here because the LinEx loss that is used to compute the score and that should thus serve as an objective is asymmetric: hence we prefer an overestimated prediction to an underestimated one. We also tried to predict the maximum between values at W-1, W-2 and W-3 in a time window of 1.5 or 2.5 hours. The second one gave an error of 29.10 on the leaderboard, and of 1.74 once overestimated.

Finally, our best prediction using this kind of simple models was obtained by correcting the evolution from one week to the next one. We first took the maximum of values at W-1, W-2, and W-3. Then, we multiplied it by what we will call the "evolution rate", that is computed as follows:

$$\text{evolution\_rate(day)} = \max\left(\frac{n_{\text{calls}(W-1,\text{day})}}{n_{\text{calls}(W-2,\text{day})}}, \frac{n_{\text{calls}(W-2,\text{day})}}{n_{\text{calls}(W-3,\text{day})}}\right),$$

if the number of calls are non-zero, with $n_{\text{calls}}(W_{-i}, \text{day})$ the total number of calls i weeks before the day for which we are predicting (sum over all time slots). Otherwise, we set evolution_rate to 1 (which boils down to not correcting the value predicted by the maximum of values at W-1, W-2, W-3).

Our best prediction without overestimating was made taking the maximum of number of calls at W-1, W-2, W-3 in a time window of 1h30 (maximum out of 9 values) and multiplying it by evolution_rate (error = 11).

5 Ensemble Methods

In this section we will describe how we used ensemble methods for our predictions, which turned out to yield the best score.

5.1 General Training and Testing Procedure

Algorithm 1 describes the general procedure to output the predictions for the submission data set, as well as the local testing procedure that allows us to compare our algorithms locally. Since the trends and the seasonal patterns in are very dissimilar depending on the assignment, we chose to train a different model for each assignment.

Moreover, the weeks in the submission data set are distributed over each month of the year 2013; and for each week for which we have to predict the number of received calls, we can only use information from the past. In order to integrate this constraint while keeping as much training data as possible, we build a different training data set for each one of these weeks, containing all the data before the considered week. We then write the predictions into the training data to use them for the following submission weeks. We use the week after the considered week in order to evaluate the error of the model, and as a hold-out data set in order to estimate the hyper-parameters of the regressors that we use.

We should note that the testing procedure that we use will yield an overly pessimistic error, since we compute the error for a given week using training data up to two weeks before; but our goal is to over estimate the error that we get in the leaderboard.

```
Result: Predictions, Error
Extract features;
for assignment in submission-assignments do
    for week in submission-weeks do
        $set_{training}, set_{submission}, set_{test} = getTrainTest(assignment, week)$;
        fit model($set_{training}$) ;
        $prediction_{submission}$ = model.predict($set_{submission}$) ;
        $prediction_{test}$ = model.predict($set_{test}$) ;
        $error_{week}$ = LinEx($prediction_{test}, ncalls_{test}$) ;
        write $prediction_{submission}$ to data ;
    end
    $error_{assignment}$ = mean($error_{week}$);
end
```

Algorithm 1: Training and Testing Procedure

5.2 Features

- Time features: We have decomposed the time slot into different features: the day, the hour, the minute (binary feature), the year and the week day, for which we use its dummy representation.
- Holiday features: We have constructed two features from the holidays: a feature indicating whether a given day is a working day right after a holiday, in order to account for the peak of calls that we have observed. We have also added a feature indicating whether a given day is a working day right before a holiday.
- Features containing the number of calls one week, two weeks, and three weeks before a given time slot for a given assignment. The idea to use these features was drawn from the fairly good performance of the simple models which use the same information. However, we have realized that neither these features nor aggregates of them improve the performance of ensemble models.

5.3 Regressors

We have tried both Random Forests and Gradient Tree Boosting models. Since the weak learners in both models are regression trees, they cannot predict values that are not in the range of values in the training data, which is very inconvenient since we have significant upward trends in some assignments. To mitigate this problem, we multiply the prediction by a an overestimation coefficient that we estimate by using the hold-out set.
Since we train a model for each pair (submission week, assignment), the training time for our global model can be very long, especially if we use a large number of estimators. Therefore, we have also implemented a parallel version of 1 that handles several weeks at the same time that allows us to have a more development process.
Our best model was obtained using Gradient Tree Boosting models with 1500 estimators, and a coefficient of 1.8, and has an error of 0.506 on the leaderboard.

6 Interpretation of Obtained Scores

The LinEx loss is defined so as to penalize underestimation more strongly than overestimation. It is defined as:

$$\mathrm{LinEx}(y, \hat{y}) = \exp(\alpha * (y - \hat{y})) - \alpha * (y - \hat{y}) - 1$$

Let us shortly analyze what a LinEx score tells us. The score is the mean of $\mathrm{LinEx}(y, \hat{y})$ for all lines in the submission file. Let us consider three artificial cases.

- Error is uniformly distributed between all predictions
- Error is concentrated in a single assignment (uniform distribution)
- Error is concentrated in a single assignment, on four weeks (uniform distribution)

Table 1: How to interpret errors

Error	Type	0.2	0.5	1	10	100	1000
Equi-distributed	Under.	5.73	8.57	11.46	26.10	46.60	69.15
	Over.	7.02	11.98	18.41	110	1,010	10,010
Dist. on one assignment	Under.	21.18	28.22	34.14	55.85	78.66	101.6
	Over.	61.97	140	270	2,610	26,010	260,010
Dist. on 4 weeks of one assignment	Under.	29.74	37.79	44.23	66.69	89.63	112
	Over.	166	400	780	7,810	78,010	780,010

Computing scores locally on a hold-out dataset by couple (assignment, week), we noticed that the error was concentrated on certain assignments (mostly Téléphonie, Tech. AXA and CAT), while the prediction was really good on some other assignments (with an error between 10^{-1} and 10^{-5}).

The assignment we identified as "hardest to predict" was Téléphonie (and it is pretty logical considering the evolution of Téléphonie, cf. Figure 6). For this assignment, the error computed on the hold-out dataset exploded for some weeks, meaning that we are not very good at predicting the sudden rise. Looking at the last line of Table 1, it would mean that if we approximately underestimate by 50 an equivalent of 4 weeks of "Téléphonie" (which is a perfectly reasonable duration) we would get a score of 0.5, approximately our final score. In reality, we probably do not underestimate a lot, as we multiply by an overestimation-dedicated coefficient, but by doing so, we overestimate on a lot of other assignments and dates. Our best score without any multiplication by an overestimation-dedicated coefficient is 11, and it approximately corresponds to underestimating by 60 on one assignment (like "Téléphonie" for example) and 4 weeks.

7 Conclusion and Perspectives

Our best model was obtained using Gradient Tree Boosting models with $1,500$ estimators, and a coefficient of 1.8, and has an error of 0.506 on the leaderboard. Note that only considering results obtained without uniformly overestimating, we reached a lower error with a hand-made predictor (cf. 4.2) than with Gradient Tree Boosting.

Ideas that we could have developed, are:

- Using auto-regressive models as ARIMA: even though we suspect that it will need further refinements, as seasonal decomposition yields a high residual, and because we have multiple seasonality. (We tried SARIMAX but encountered technical difficulties in building an automatic method to select the hyper-parameters.)
- Focus more on the assignments which we know yield an unsatisfying score: "Téléphonie", "Tech AXA", "CAT", with methods able to predict a trend.