Machine Learning Project

Nathan Rouxel, Robin Monnier, Victor Challier and Emilien Garreau
M2 Data Sciences

January 12, 2017

Abstract

balablabalabalbalb

Contents

1. Example

This state of the art is quite good [1]

2. Data analysis

In order to have some insight about our data, we study the range taken by the values and the different values taken (see .1):

It is pretty obvious from Figure .1 that errors in the category "Telephonie" will be at the source of most of the global error. This is why we decided to focus first on this category.

A first plot allow us to see how fast the number of calls can evolve. We can see in Figure 2.1 that the evolution of the total number of calls is mostly due to the "Telephonie" category.

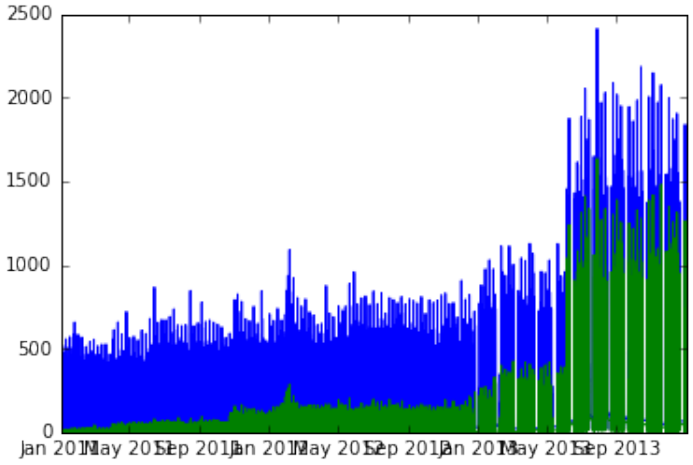


Fig. 2.1: *Number of calls in the "Telephonie" category (green) and total number of calls (blue)*

We observe that the number of calls depends heavily on whether the day is a business day or not (see Figure .2).

3. Preprocessing

In order to simplify the problem and after studying all the features, we decided to select only the date and the number of calls as base features. So we need to preprocess the csv file to extract those. The preprocessing is done by our class **Preprocessor**. This step runs in two parts.

3.1. Get all the categorical features

We can classify the features of the dataset in two parts the numeric features and the categorical features. What we mean by categorical features

3.2. Create a valid dataset

After dataset exploration we quickly realised, that the data wasn't sorted at all which leads to the first step sorting all our data on the datetime (column number 1). This step has been done using Linux functionalities:

Code 1: *Sort the csv file on the date*

```
head -1 train.csv > sortTrain.csv\;
tail -n+2 train.csv | sort --parallel=8 -t ';' -k1 >> sortTrain.csv
```

When this step is done we want to create a numpy array of all the features which is quicker to load in order to train our model:

Result: Return X a numpy array of
x = 0;
// We cannot load all the dataset on our machine we need to use chunk
for *Each chunk (dataframe) of train.csv* **do**
  if *not first chunk* **then**
    new_df = concat tmp_df with chunk;
  end
  get the minimum date of new_df;
  get the maximum date of new_df;
  // We assume that for the maximum date we haven't all the information in this chunk
  tmp_df = all tuples which corresponds to the maximum date;
  df = all the other tuples;
  get the new maximum date of new_df;
  With the min and max dates create a list of all the time slot between the two dates;
  // Preprocess the data
  Group by **DATE** and **ASS_ASSIGNMENT** on **CSPL_RECEIVED_CALLS** doing a sum;
  Pivot all the value on **ASS_ASSIGNMENT** to get at each time step the specific number of calls;
  Complete the time step with unknown values with 0 and missing ASS_ASSIGNMENT in the chunk;
  x_chunk = results of the previous operation ;
  x = concatenation of x and x_chunk ;
end
save x;

Algorithm 1: Preprocessing of the csv file (you can inspect the code in the method __preprocess_csv in preprocessing.py)

4. Algorithm

4.1. Best approach

4.1.1. Feature engineering

References

[1] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014.

Appendices

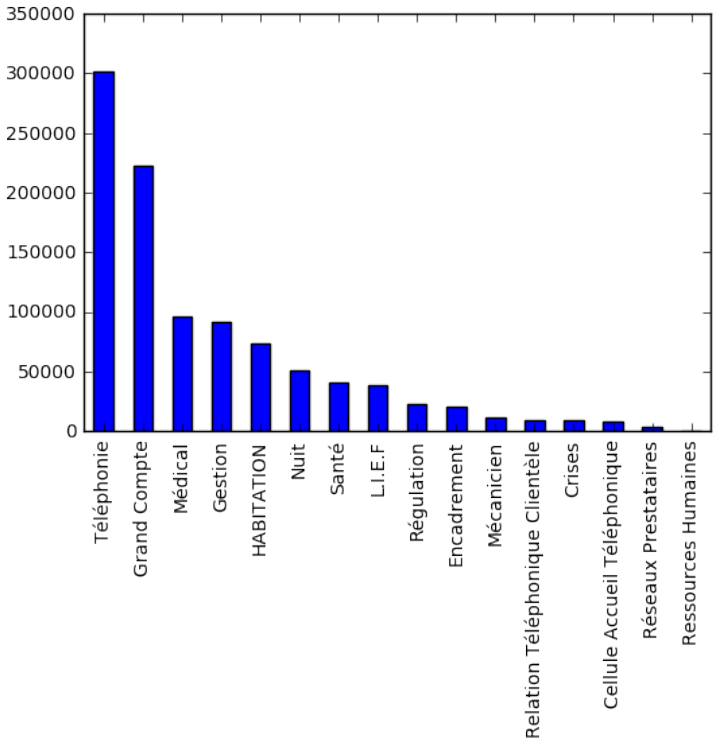


Fig. .1: *Values of the different categories to be predicted*

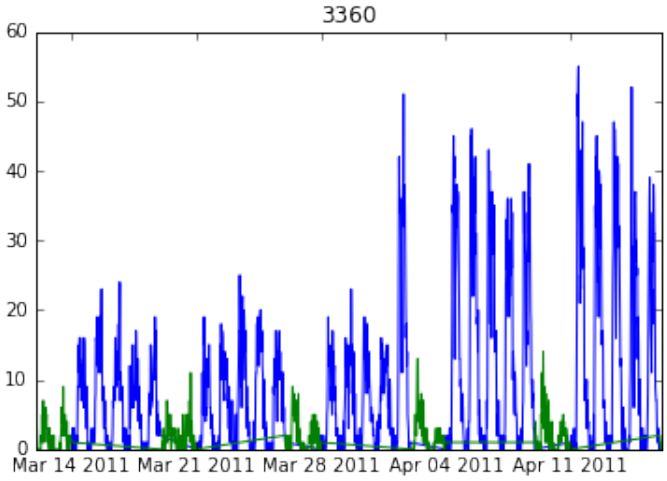


Fig. .2: *Number of calls during worked day (in blue) and holiday (in green)*

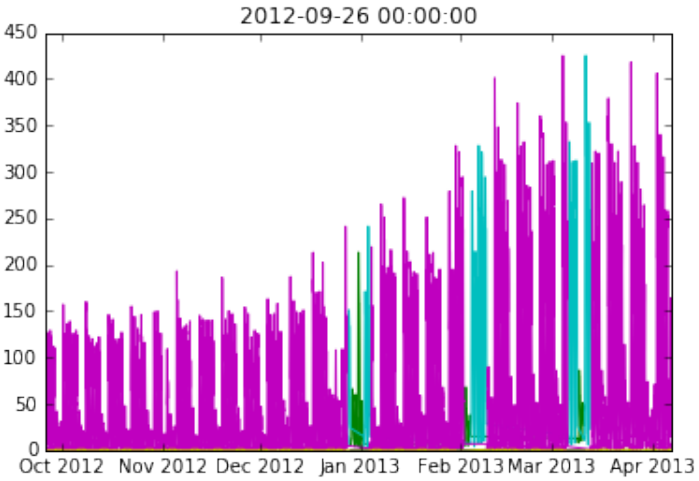
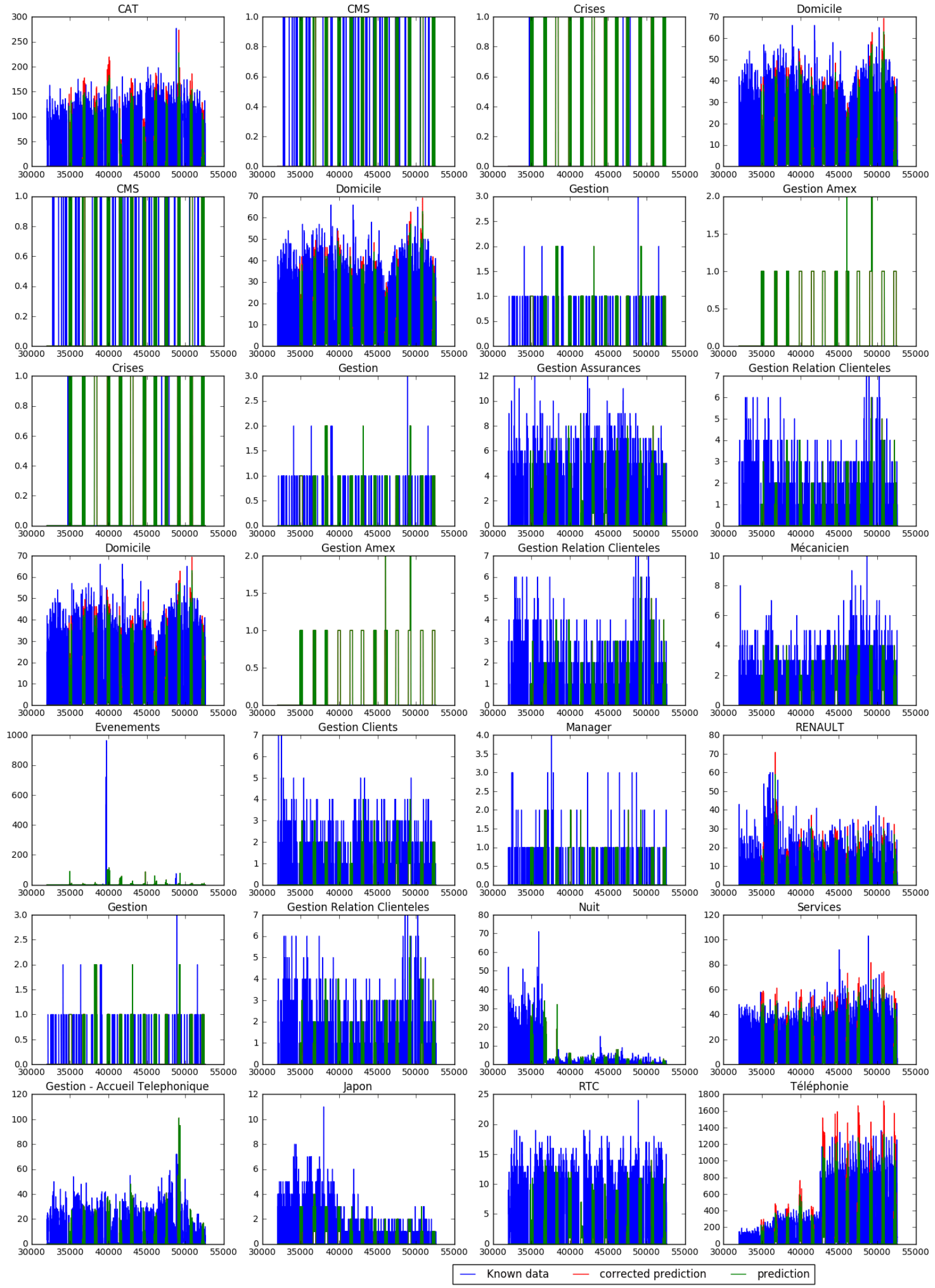


Fig. .3: *Result in the naive algorithm*

Code 2: *Create the recurrent neural network with 28 outputs*

```
X = np.concatenate((x_holidays, x, mean_over_time), axis=1)
_, nb_features = X.shape

# Model creation
batch = 336
main_input = Input(batch_shape=(batch, 1, nb_features), name='main_input')
lstm = LSTM(32, stateful=True)(main_input)
drop = Dropout(0.5)(lstm)
outputs = []
for i in range(28):
    outputs.append(Dense(1, name='out{0}'.format(i))(drop))
model = Model(input=main_input, output=outputs)
model.compile(optimizer='adam', loss='mean_squared_error')
model.load_weights('../dataChallenge/weights/stateful/model_36624_36959_loss_2.5011280818.h5')
```


CAT
CMS
Crises
Domicile
CMS
Domicile
Gestion
Gestion Amex
Crises
Gestion
Gestion Assurances
Gestion Relation Clienteles
Domicile
Gestion Amex
Gestion Relation Clienteles
Mécanicien
Evenements
Gestion Clients
Manager
RENAULT
Gestion
Gestion Relation Clienteles
Nuit
Services
Gestion - Accueil Telephonique
Japon
RTC
Téléphonie
Known data
corrected prediction
prediction