Python for Data Analysis

# Facebook Comment Volume Prediction

Hélène Peignard - Julie Mordacq
DIA 4

## Introduction

Our **dataset** was: the Facebook Comment Volume Dataset. Since several years, social networking services use has highly increased. This dataset fit in the  need to understand the behavior of users on those services. In 2016, Facebook was the most active social network with 1.76 billion users.

Thus, the **goal** is to predict the number of comments in the next H hours (H being a feature of the dataset).

The data was collected by Kamaljot Singh in 2016.

## The dataset

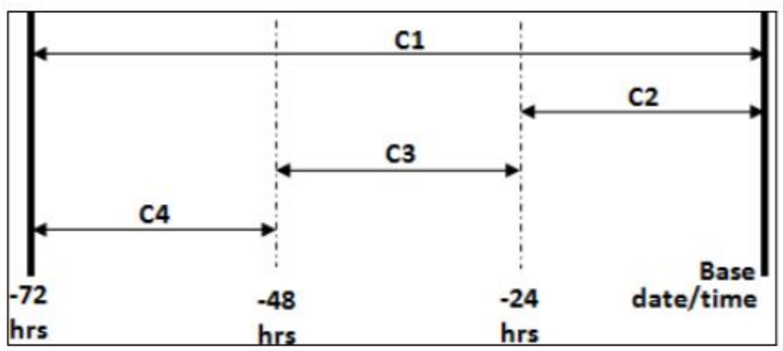
## The Features

### The dataset contains 54 features :

- ➔ **Page features** : defines users global interest in the page
  3 continuous features (*page likes*, *page checking's, page talking about)*
  1 categorical features (*page category)*
- ➔ **Essential features**: describes the pattern of comments on the post and on the page
  35 continuous features (e.g. : CC1, CC2,CC3,CC4,CC5)
- ➔ **Weekday features*:*** defines on which date the post was published and the day selected to make the prediction
  Binary indicators are used.
  14 binary features
- ➔ **Post related features**: describes the post (e.g: post length, post shares)
  5 continuous features
- ➔ **Target variable**

## Specifications on features

- ➔ Two types of time features
  **published time** : refers to the time when the post was published
  **base time**: refers to the time when we yearn to make the prediction
- ➔ Essential features



The essential features
source : Facebook Comment Volume Prediction, Kamaljot Singh

## The variants

One specificity of this dataset is: the variants. There are several files named : features_variant_X : X referencing to how many instances of the final training set is derived from a single instance.

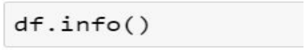
However, we didn't find a way to deal with those instances since every post is not clearly identified, and that some pages have several posts (making some instances very similar. Indeed, in this case, all the instances concerning the page are identical). Furthermore, there were already a sufficient number of data in variant 1.

Thus, we decided to work with only the first variant.

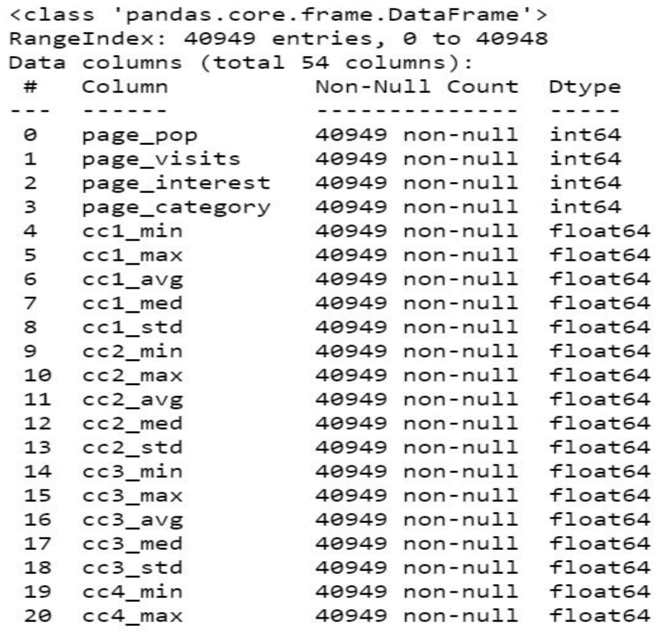
## The dataset analysis

# The Instances

We have **40949** entries,with **no missing values**.



```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40949 entries, 0 to 40948
Data columns (total 54 columns):
 #    Column          Non-Null Count  Dtype
---   ------          --------------  -----
 0    page_pop        40949 non-null  int64
 1    page_visits     40949 non-null  int64
 2    page_interest   40949 non-null  int64
 3    page_category   40949 non-null  int64
 4    cc1_min         40949 non-null  float64
 5    cc1_max         40949 non-null  float64
 6    cc1_avg         40949 non-null  float64
 7    cc1_med         40949 non-null  float64
 8    cc1_std         40949 non-null  float64
 9    cc2_min         40949 non-null  float64
 10   cc2_max         40949 non-null  float64
 11   cc2_avg         40949 non-null  float64
 12   cc2_med         40949 non-null  float64
 13   cc2_std         40949 non-null  float64
 14   cc3_min         40949 non-null  float64
 15   cc3_max         40949 non-null  float64
 16   cc3_avg         40949 non-null  float64
 17   cc3_med         40949 non-null  float64
 18   cc3_std         40949 non-null  float64
 19   cc4_min         40949 non-null  float64
 20   cc4_max         40949 non-null  float64
```

## Data visualization

We performed several data visualization, in order to understand the links and correlations between the variables, but also their importance, as well, as their scales or distributions. This enabled us to make some decision regarding the selection of the features and the global pre-processing of the data.

## The data preprocessing

## We modified, deleted and added several features, given the dataset analysis we conducted.

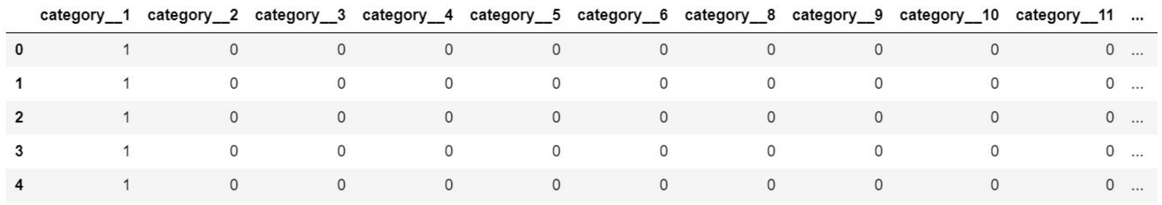
We ended up with : **126 features**.
And we standard scaled all the features.

## Post_category

post_category was a **categorical feature** and each category had no quantifiable relationship with each other, for instance 1 represented the category encoded by one being *Product/service* and 2 *Public figure*.

Categorical features represented by increasing numbers might induced **bias in models**, indeed they might be understood as a hierarchy by the model but there is none.

Thus, we decided to one-hot encode the post_category column. We transformed the column post_column in several columns (106, since they are 106 categories) taking only 1 and 0 (similar method to the weekdays column).



| | category__1 | category__2 | category__3 | category__4 | category__5 | category__6 | category__8 | category__9 | category__10 | category__11 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **1** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **2** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **3** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |
| **4** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |

## Weekdays published

According to the correlation matrix and the plot of target variable in function of weekdays, we noticed that the published day of the post had no impact on the target variable.

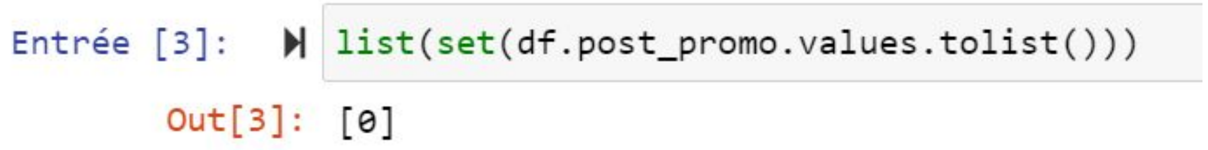
After several tries, we decided not to take into account weekdays published features.

## meanCC

We created a new feature : meanCC, which is the arithmetic  average of CC2, CC3 and CC4. (We did not take into account CC1 since it's already the sum of CC2, CC3 and CC4). This new variable allows to have an insight on whether the number of comments after the given H is significant or not on average.

## post_promo

The column post_promo is only constituted of 0. Thus, we decided to drop it since it wouldn't have any influence on the target value.



```
Entrée [3]:  list(set(df.post_promo.values.tolist()))

   Out[3]: [0]
```

## The models

## A regression problem

We want to predict how many comments a post will get given a certain hour H. Thus, this  poses as a regression problem.

We tested several models:

➔ regression trees
➔ random forest
➔ linear regression
➔ gradient boosting

## The metric

We used one metric to compare our models  apart from accuracy: the Root Mean Square Error.

**RMSE** allows  to evaluate how concentrated the data is around the line of best fit.
It is a good measure of how accurately the model predicts the response.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

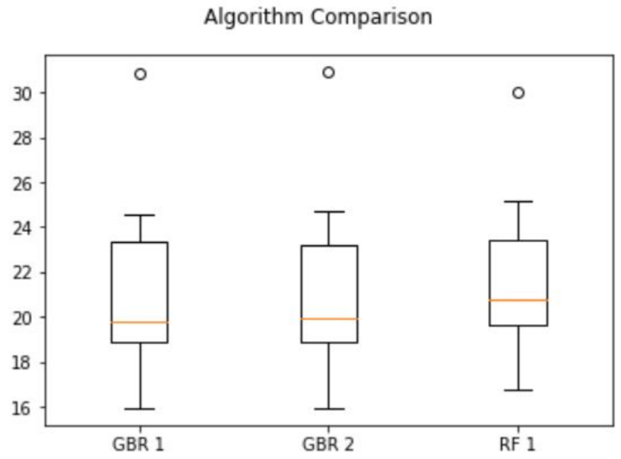
## A first glance

| Model | RMSE |
|---|---|
| Regression Trees | 28,33 |
| Random Forest | 20,88 |
| Linear Regression | 31,34 |
| Gradient Boosting | 19,68 |

At first sight, Random Forest and Gradient Boosting seem to perform better on our data.

# The final model

After performing grid search to find better hyperparameters, and cross-validation to find the best suitable model we decided to select **Gradient Boosting** which has a slightly better RMSE



## API and Interface

## The API

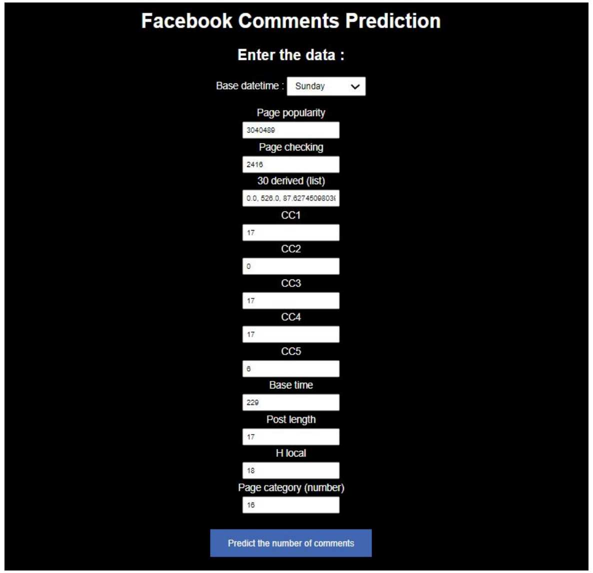
To turn the model in the notebook into an API, we decided to use the Pickle module, allowing us to make it into a file that can then be reused on the interface easily, to predict new values.

That way, we only had to load it into the code for the interface.

We ended up having 2 pickle files, 1 for the model and 1 for the scaling, since the value that will be entered on the interface will need to be scaled to make a prediction based on them.

# The interface

The interface has been made using Flask. It has 2 pages, the first  one being used to input new values, and then when the prediction has been made, the user is redirected to another page to see them.



**Facebook Comments Prediction**

**Enter the data :**

Base datetime : Sunday

Page popularity: 3040489

Page checking: 2416

30 derived (list): 0.0, 526.0, 87.62745098039

CC1: 17

CC2: 0

CC3: 17

CC4: 17

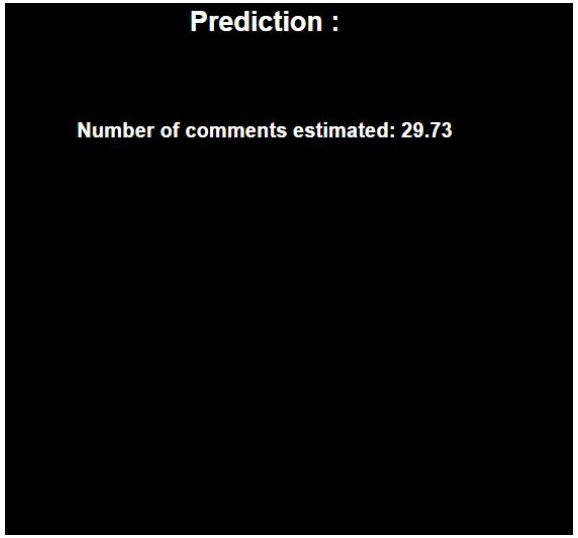
CC5: 6

Base time: 229

Post length: 17

H local: 18

Page category (number): 16



**Prediction :**

**Number of comments estimated: 29.73**