Facebook Comment Volume Prediction Dataset

Léa Wong - Python for Data Analysis - 2021

Objectif

Predict the number of comments for a Facebook post for the X next hours

Steps

1. Presentation of Dataset
2. Inspection/Cleaning of data
3. Visualization of data
4. Machine Learning

Dataset

Comments picked on several Facebook pages

Data are pre-treated:

—> no comments older than 3 days before de data collect

—> no incomplete data

Dataset split into 5 files corresponding to the number of data collecting base time.

Base Time

Data are collected based on a X multiplier (from 1 to 5).

For each variant, we find X instances of each post with different base times.

Since the number of final comment is known, the author wanted to add difficulty, by adding the ‘hrs’ variable in my code, corresponding to a random integer between 1 and 24. This indicates the time gap between the base time and the prediction.

Inspection of data

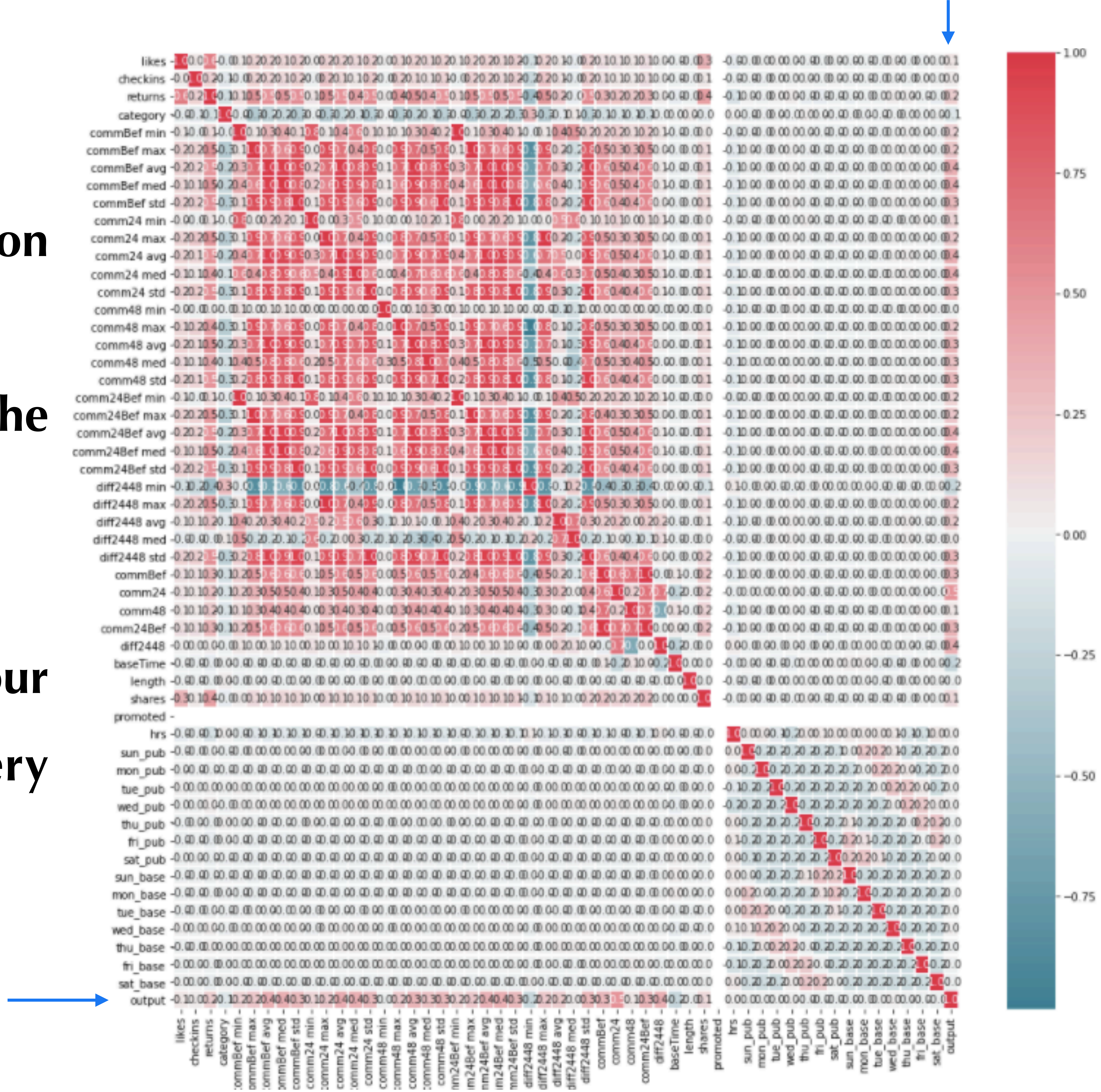
Each variant possesses 53 numerical variables and 1 target.

diff2448 max	diff2448 avg	diff2448 std	commBef	comm24	comm24Bef	diff2448	baseTime	promoted	hrs	sun_pub	mon_pub	tue_pub	wed_pub	thu_pub	fri_pub	sat_pub
806.0	4.970149	69.85058	0	0	0	0	65	0	24	0	0	0	1	0	0	0
806.0	4.970149	69.85058	0	0	0	0	10	0	24	0	0	0	0	1	0	0
806.0	4.970149	69.85058	0	0	0	0	14	0	24	0	0	0	0	0	1	0
806.0	4.970149	69.85058	7	0	7	-3	62	0	24	0	0	0	0	0	1	0
806.0	4.970149	69.85058	1	0	1	0	58	0	24	0	1	0	0	0	0	0
806.0	4.970149	69.85058	0	0	0	0	60	0	24	0	0	1	0	0	0	0
806.0	4.970149	69.85058	0	0	0	0	68	0	24	0	0	0	1	0	0	0
806.0	4.970149	69.85058	1	0	1	-1	32	0	24	0	0	0	0	1	0	0
806.0	4.970149	69.85058	0	0	0	0	35	0	24	0	0	0	0	0	1	0
806.0	4.970149	69.85058	0	0	0	0	48	0	24	0	0	0	0	0	1	0

Here is an overview of the correlation matrix obtained.

The output target is resented on the last row and the last column.

We easily see thanks to the colour scale that the output is not very correlated with other variables.



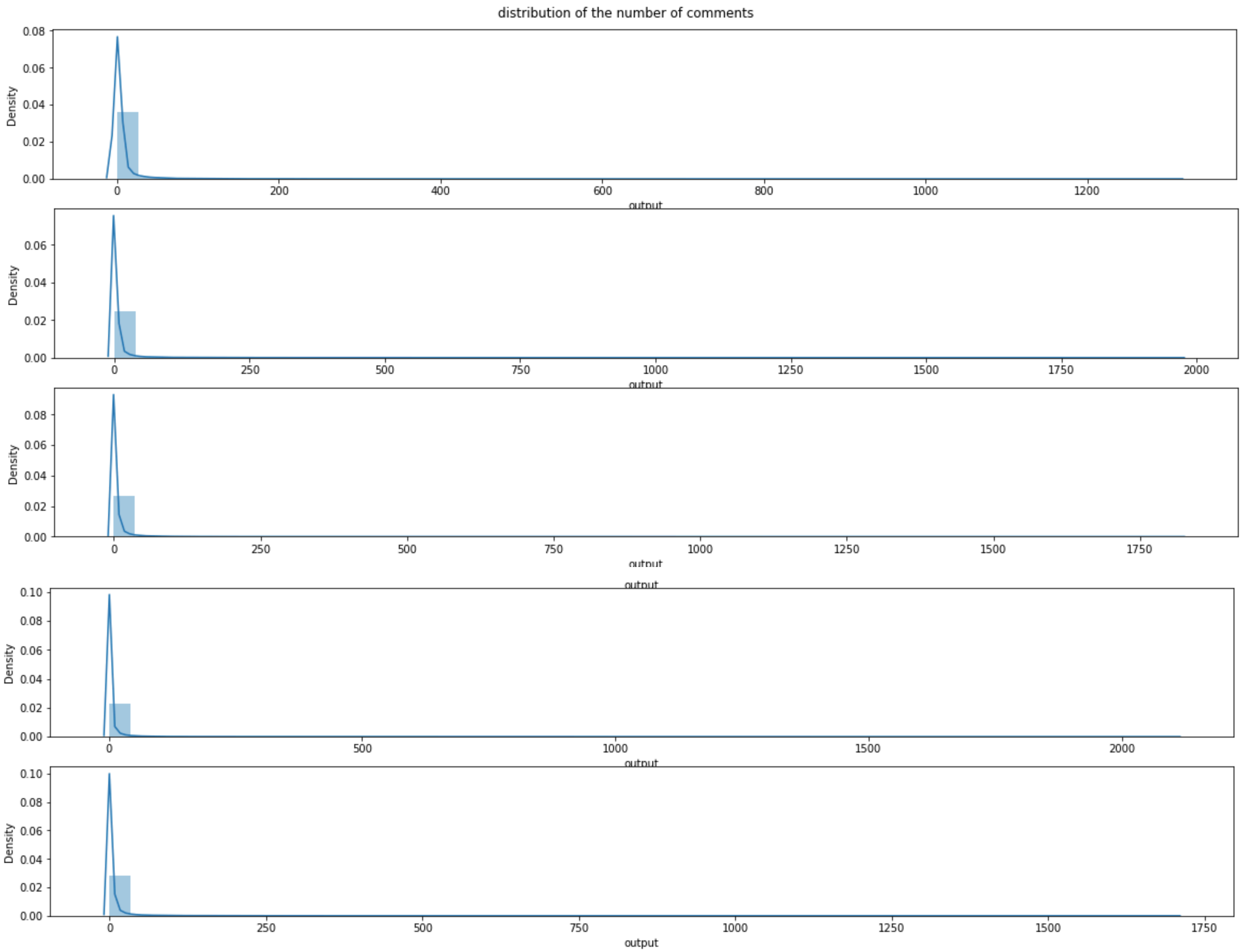
Data cleaning

Cleaning of less influential parameters (correlation $\in$ [-1.9 ; 1.9])

Withdrawn parameters : 'checkins', 'comm24 min', 'comm24Bef min', 'comm48', 'comm48 min', 'commBef min', 'diff2448 med', 'length', 'likes', 'returns', 'shares'

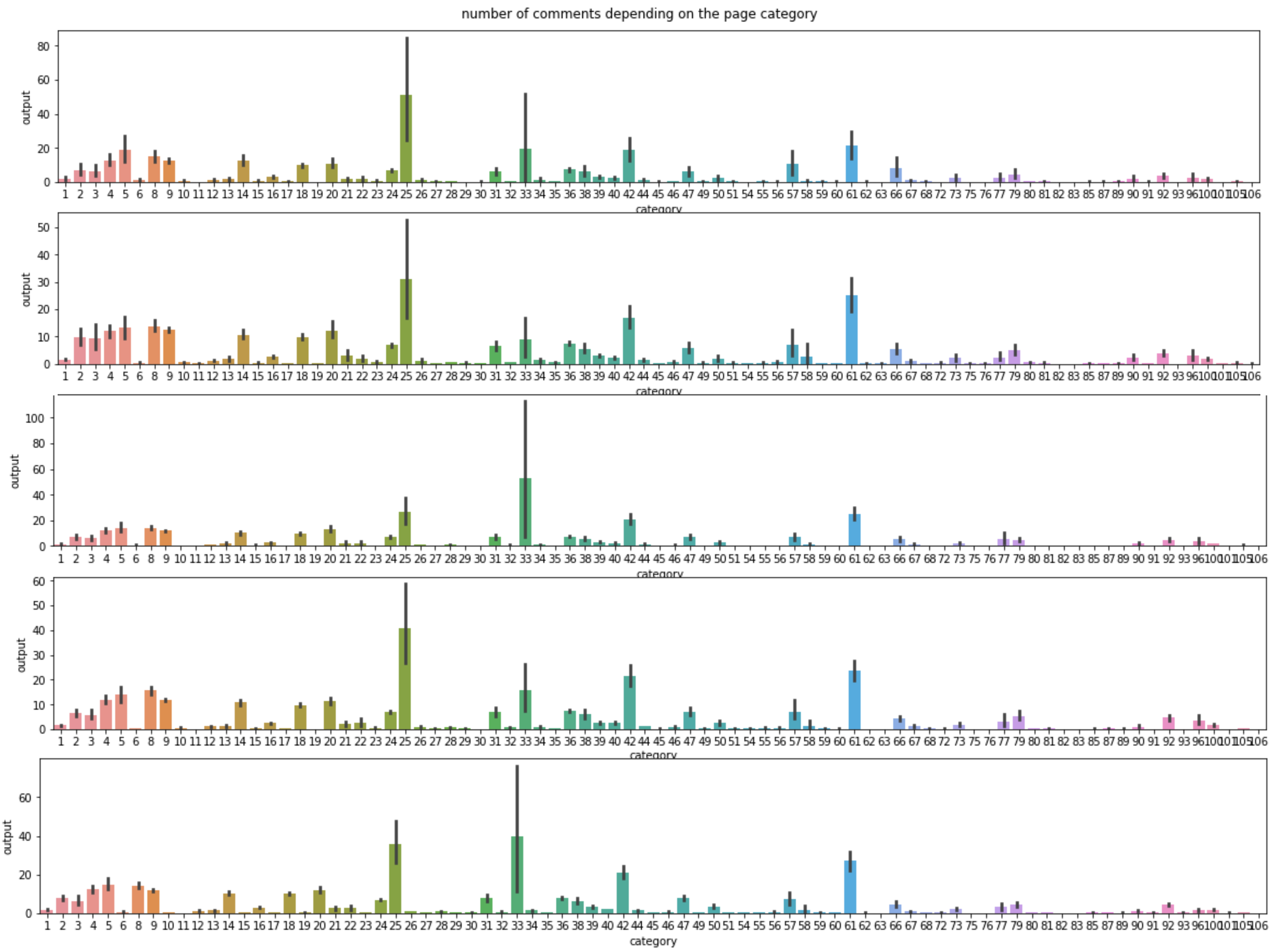
—> 53 => 42 parameters

Data Visualization



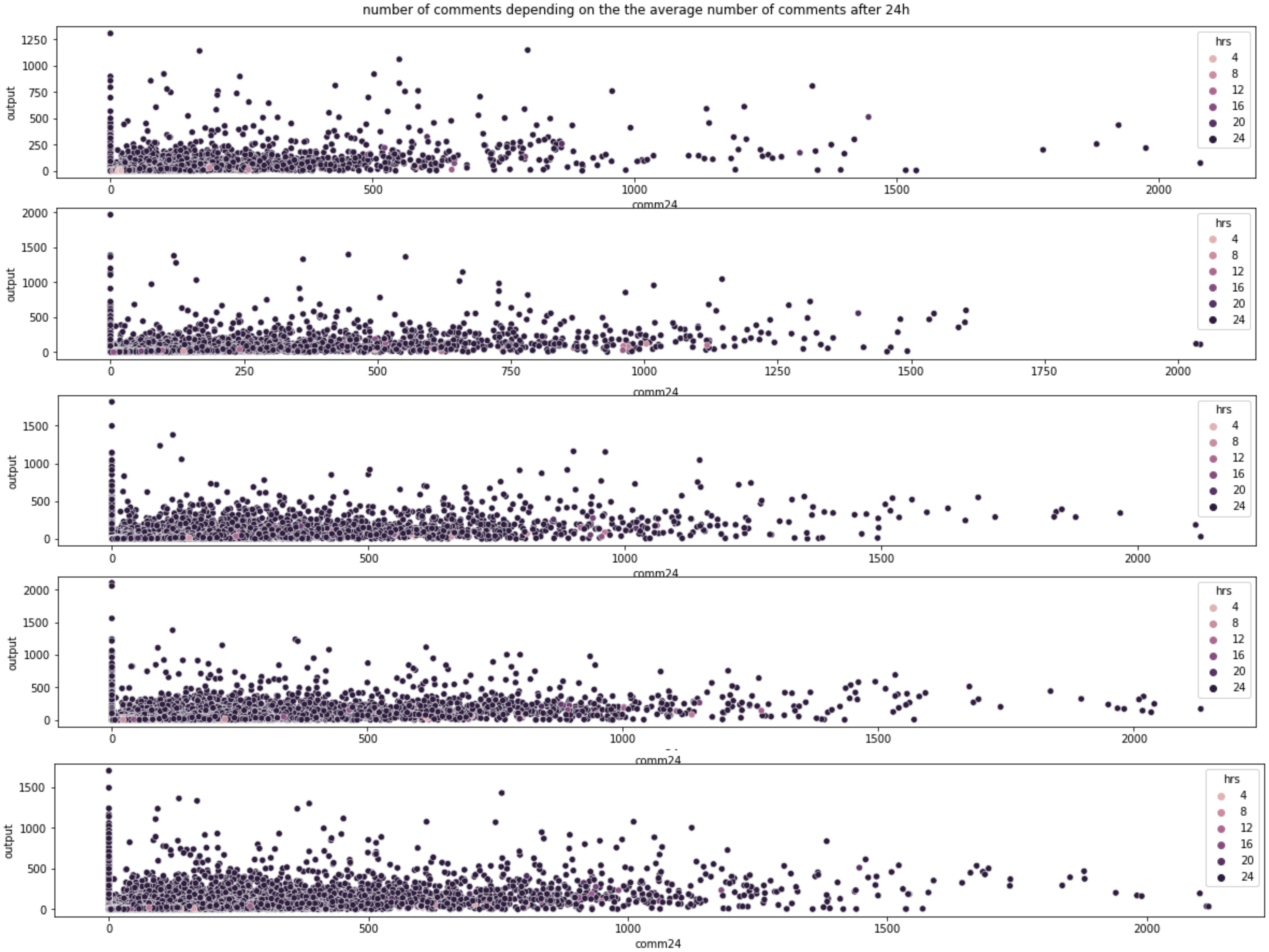
First, let's see the distribution of the target value.

As we can see, most of the data are null or very low (-100)



Here, we find the number of comments depending on the page category.

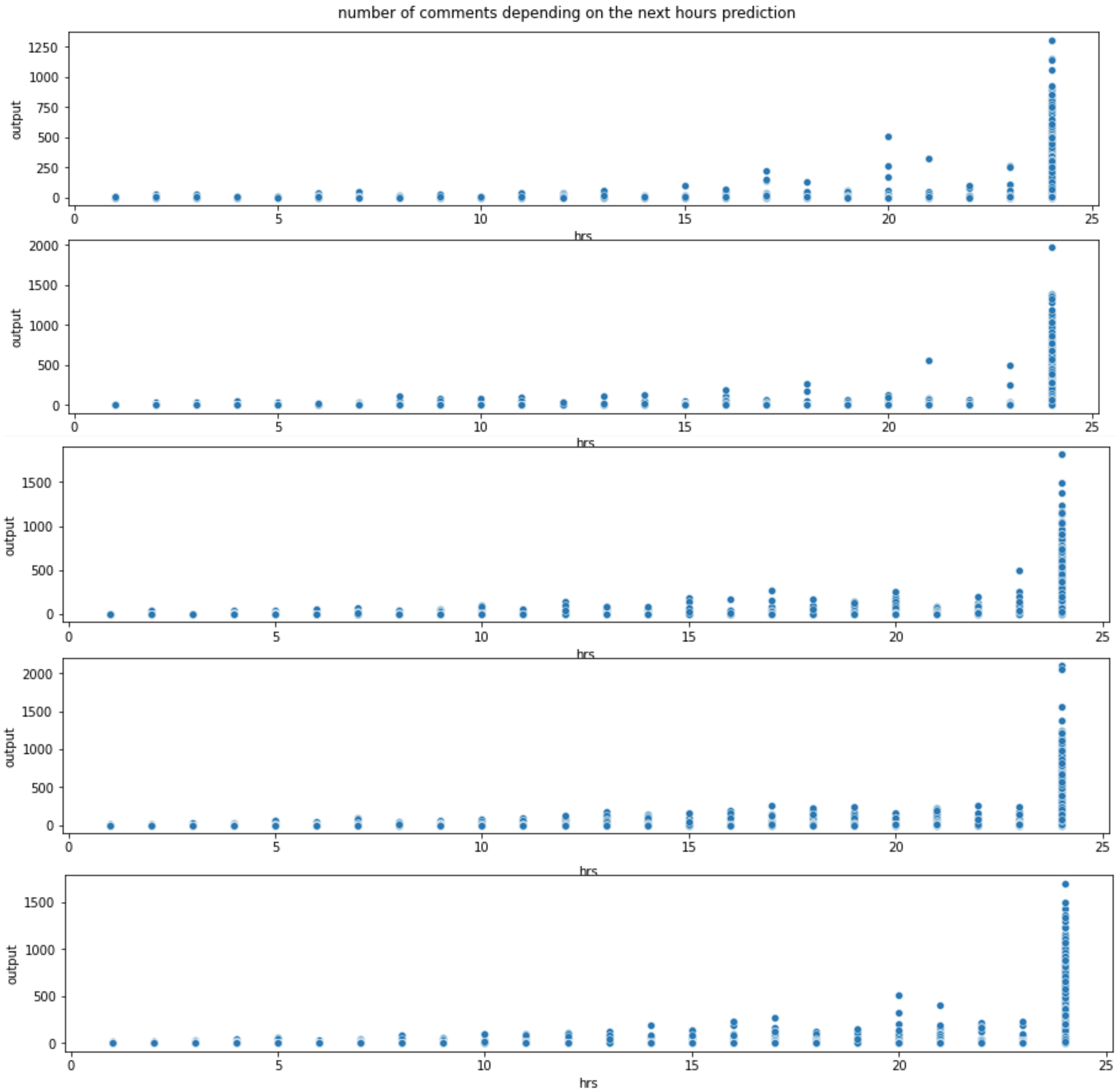
We still notice that some categories stand out. That's why I decided to keep this parameter for the predictions.



Here, we find the number of comments depending on the number of comments already present 24h before the associated base time.

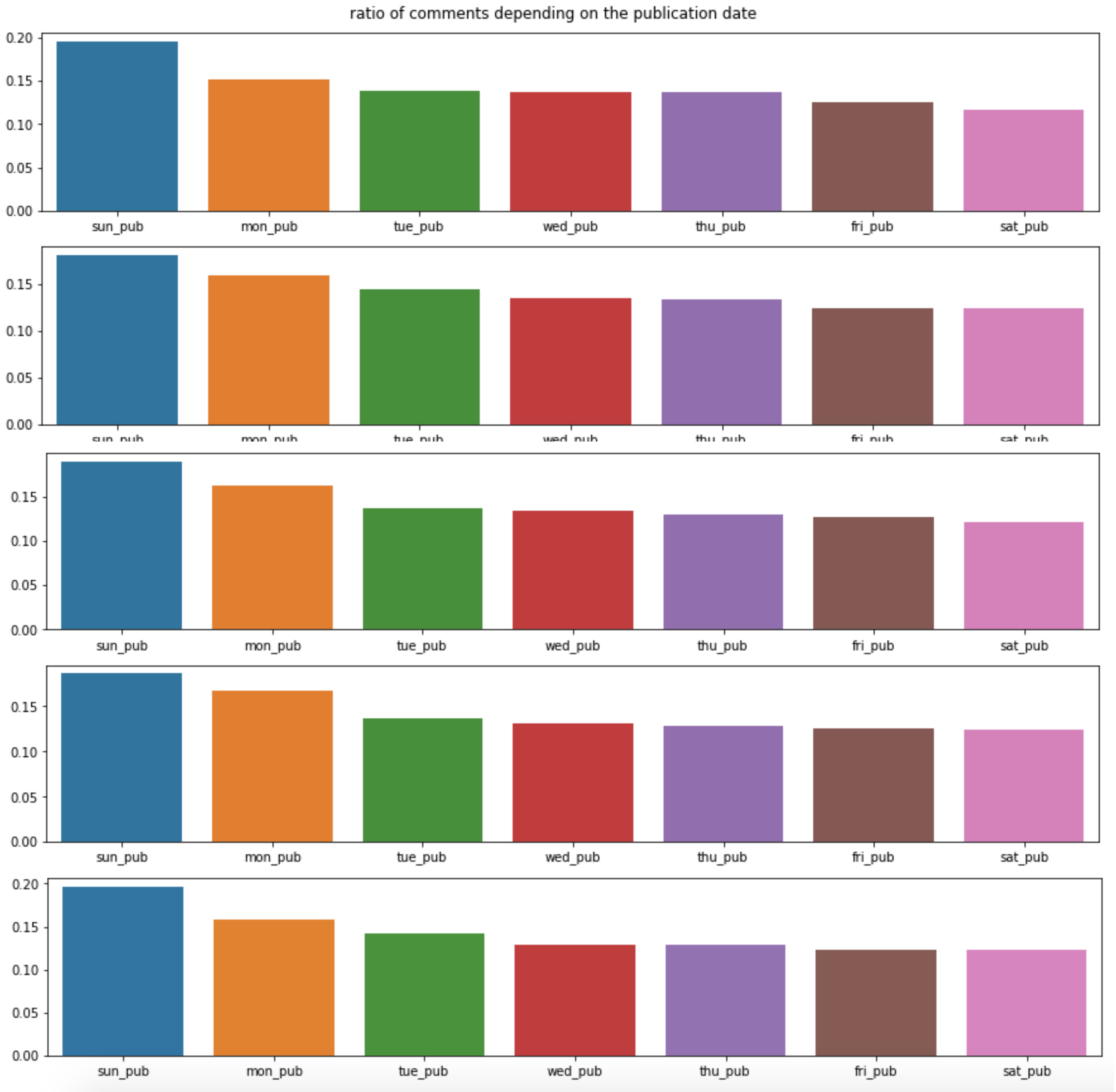
I added the hour gap between the base time and the prediction with the hue parameter.

Most of the prediction are made 24h after the base time. Even if it is the most influential variable, we can't seem to see the link.



To confirm the tendency of predictions mostly asked 24h after the base time, I plotted the graphics depending on the variable 'hrs'.

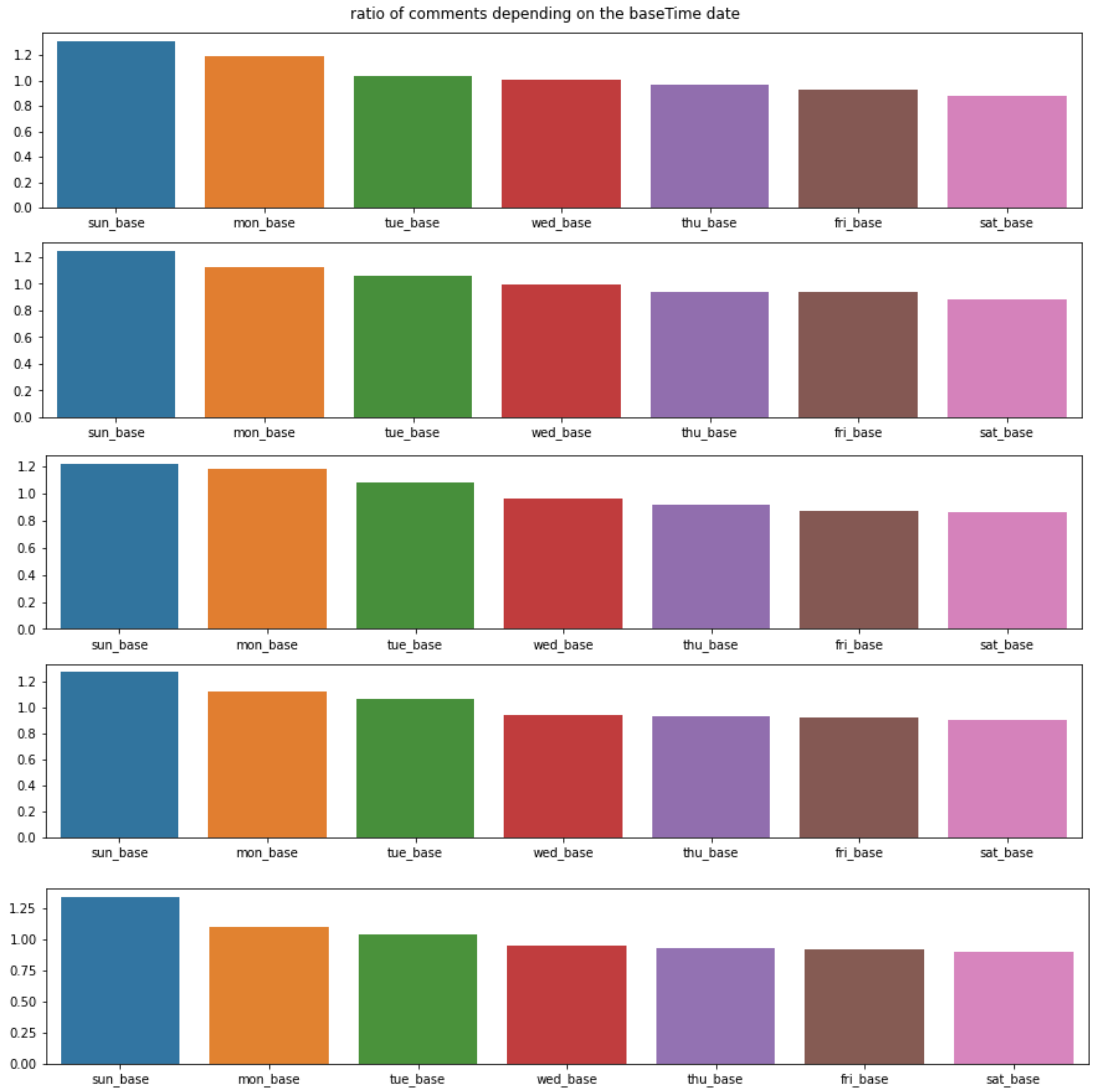
As we suspect, post receive more comments during a longer time period. Moreover, it confirms that 24h is the most common period within the dataset.



The histogram opposite represents the days during which the various posts were published.

To have a more precise idea, I made a ratio of the number of comments compared to the total number of comments.

You can notice that it's for those published on a Sunday, and generally at the beginning of the week that we have the most comments.



We find the same pattern for the day of the baseTime chosen to collect the data.

These results are not surprising because comments are predicted for a maximum of 24 hours after publication.

Machine Learning

ESILV ENGINEERING SCHOOL DE VINCI PARIS

Selected models :

- **Linear Regression**
- **Random Forest**
- **Decision Tree**
- **Elastic Net**

Best Parameters :

- **Linear Regression :**

‘fit_intercept’ : False, ‘normalize’ : True

- **Random Forest :**

‘max_depth’ : 10, ‘max_features’ : 'auto', ’n_estimators’ : 50

- **Decision Tree :**

‘max_depth’ : 6, ‘criterion’ : ‘friedman_mse', ‘max_features’ : 'auto'

- **Elastic Net :**

‘alpha’ : 6, ‘l1_ratio’ : 6

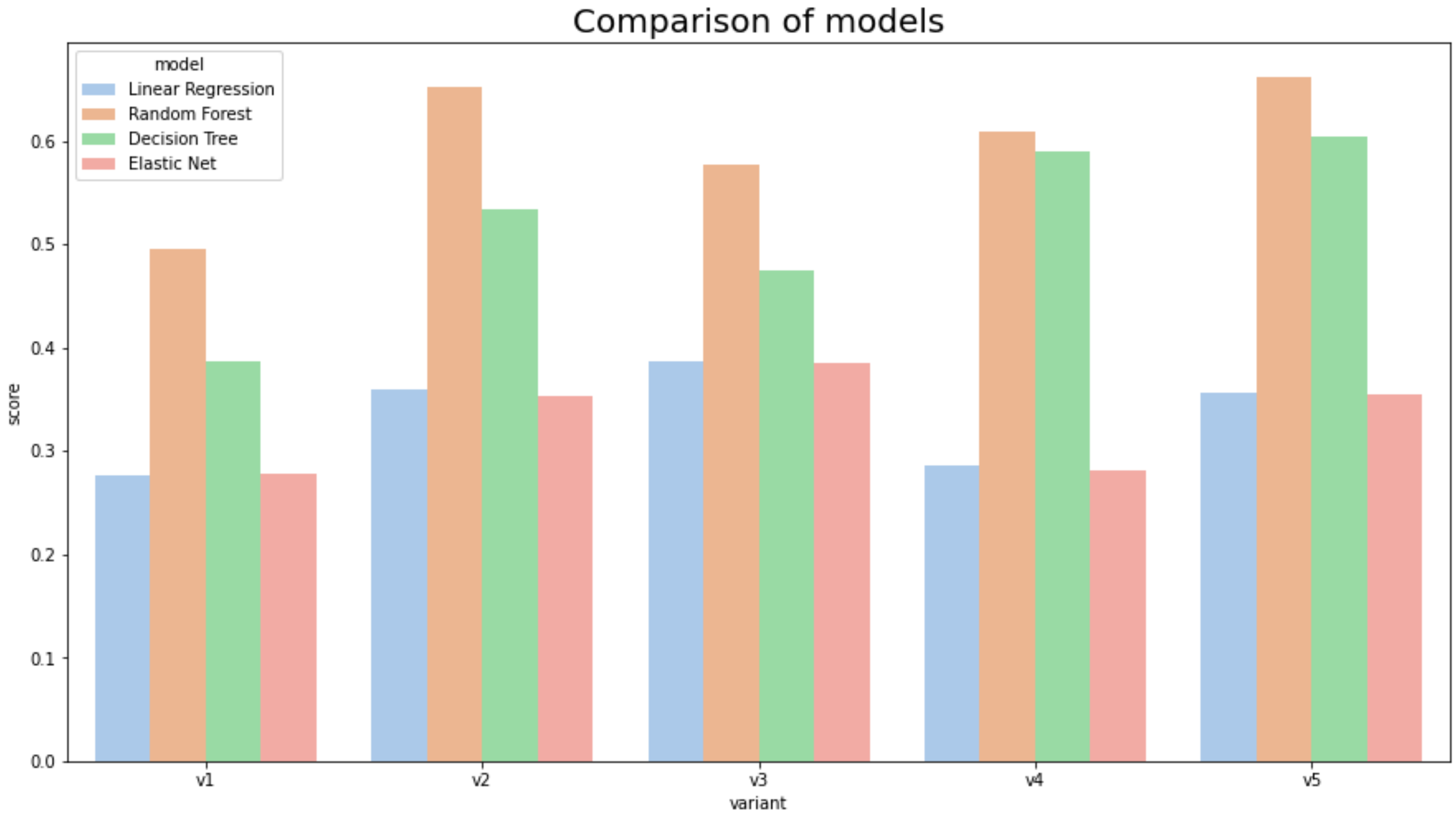
Optimization of hyper parameters using a search grid for all models.

Results of the models

	variant	model	score
0	v1	Linear Regression	0.275785
1	v2	Linear Regression	0.360291
2	v3	Linear Regression	0.386607
3	v4	Linear Regression	0.285650
4	v5	Linear Regression	0.355495
5	v1	Random Forest	0.495624
6	v2	Random Forest	0.652346
7	v3	Random Forest	0.577130
8	v4	Random Forest	0.609556

	variant	model	score
14	v5	Decision Tree	0.603665
15	v1	Elastic Net	0.277286
16	v2	Elastic Net	0.353298
17	v3	Elastic Net	0.384892
18	v4	Elastic Net	0.281827
19	v5	Elastic Net	0.354332

Comparison of the models



For all our variants, the most efficient model is the Random Forest (accuracy between 49% and 66%).

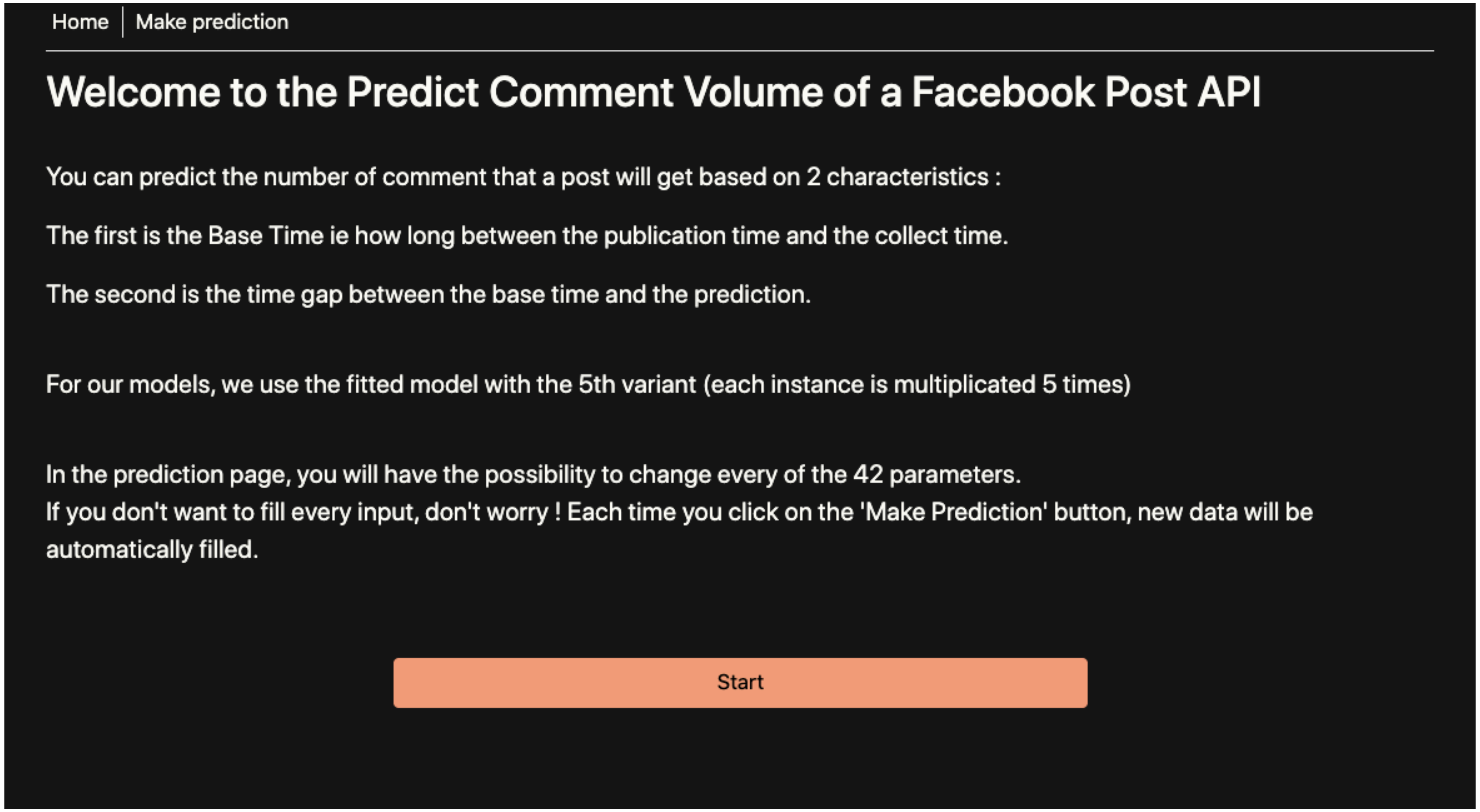
followed by the Decision Tree (accuracy between 38% and 60%)

Finally, Linear Regression and Elastic Net have similar results (accuracy between 27% and 38%).

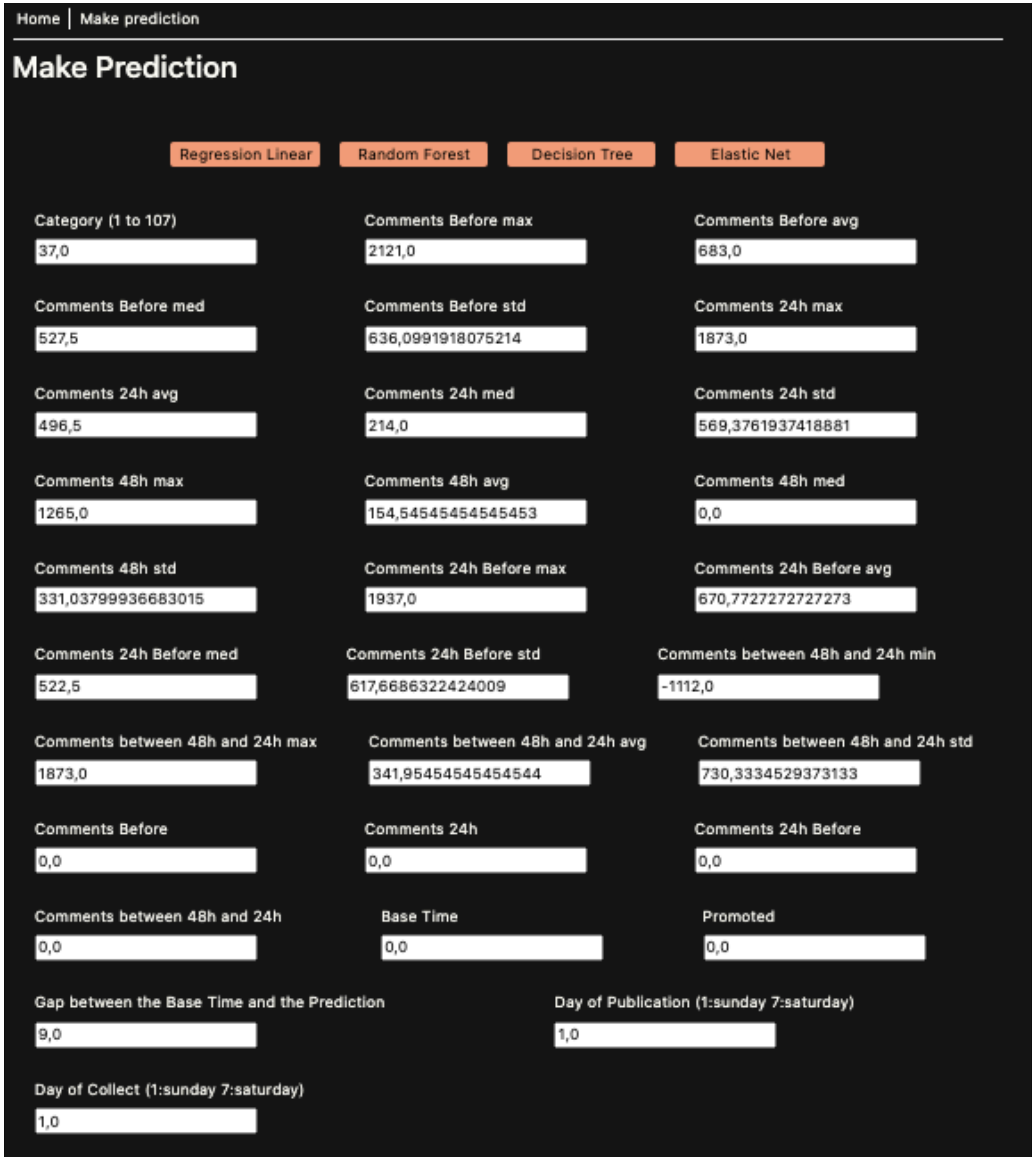
API with Flask

For the API, I decided to save the models fit with the 5th variant since it's the variant with the most accuracy since it was trained with more data.

Moreover, all variant have the same parameters so it was useless to use all of them.

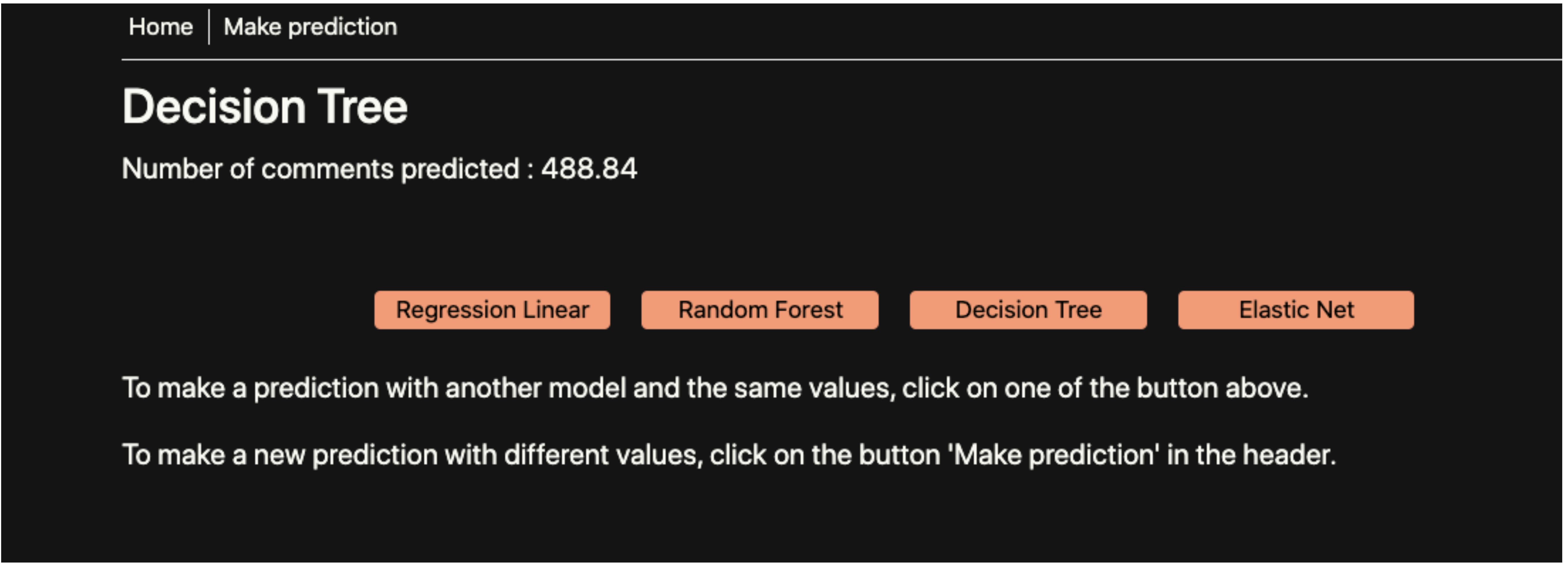


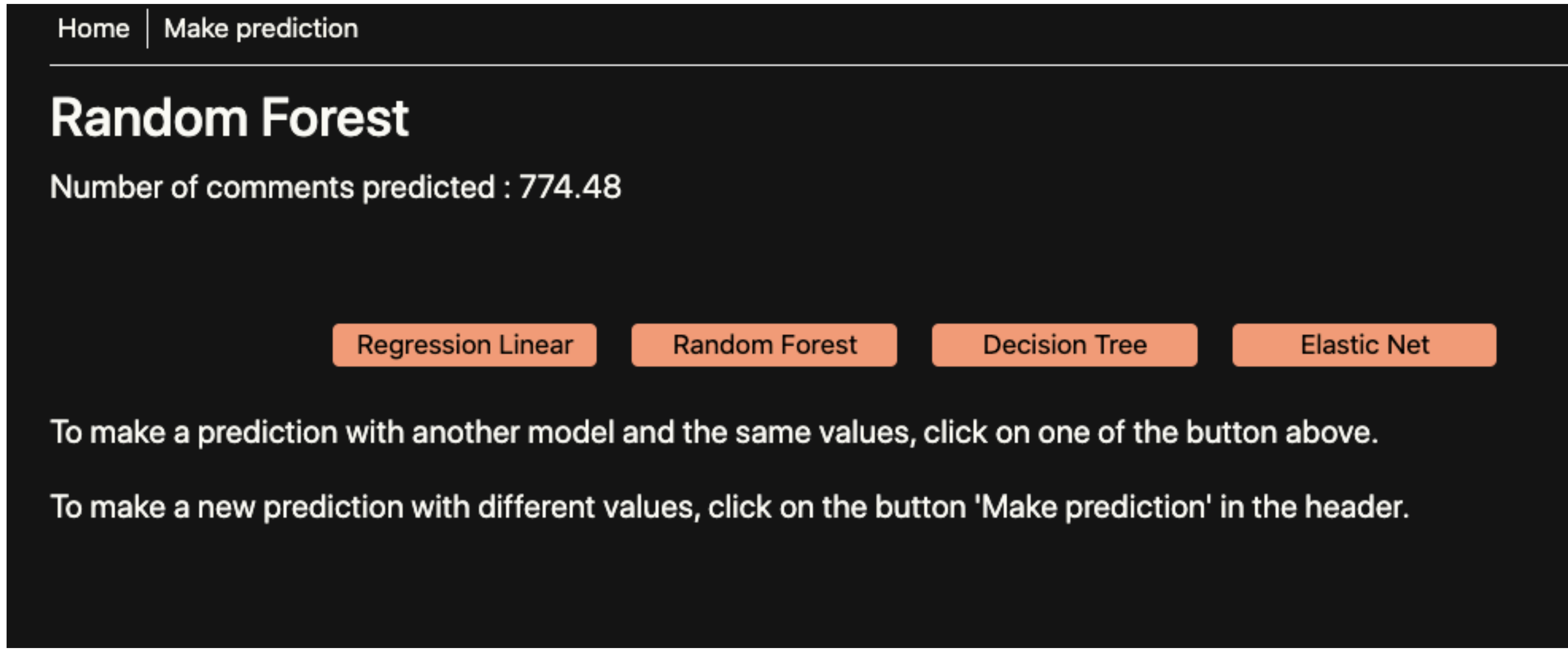
API with Flask



Here the user can change the pre-filled data

API with Flask





We can see the different results obtained with all our models with the same values