I - Exploration and EDA

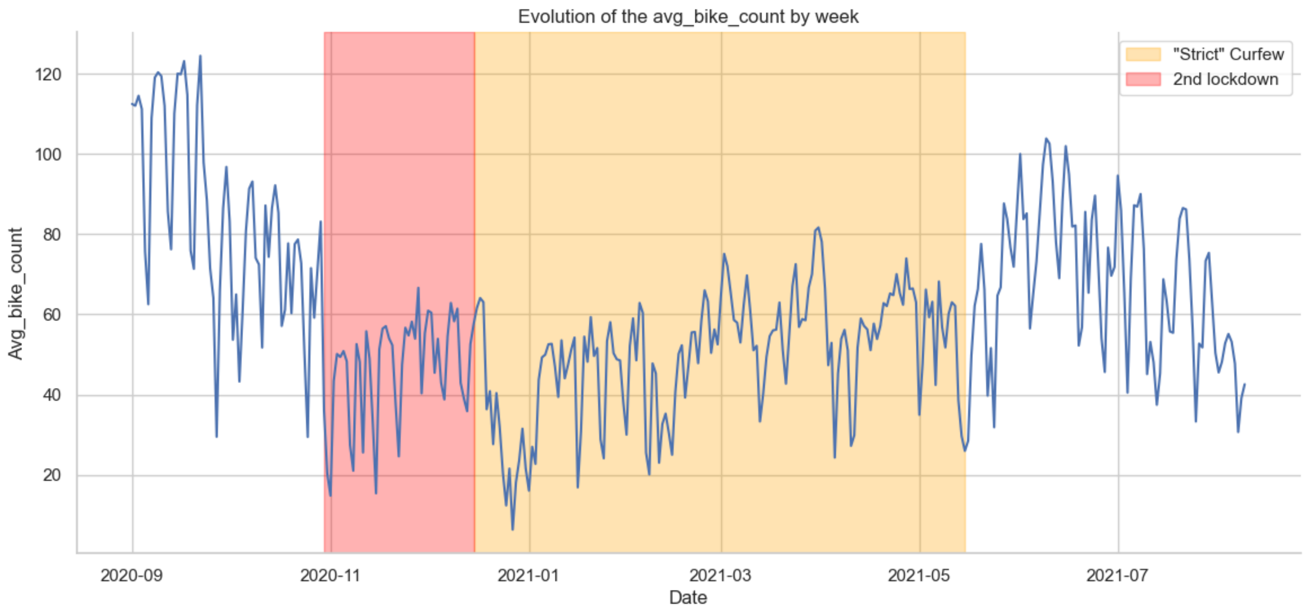
a) Temporal aspect of the data

The first thing we noticed upon inspection of the dataset was that our project was a **Time Series analysis**. Indeed, thanks to the starting kit kindly provided by the teachers, we knew that the cyclical aspect intrinsic to the temporality of our data was going to play a crucial role in our estimator. It also enabled us to quickly grasp which bicycle counters had been chosen and why the logarithm was the target variable. The function for encoding dates in columns of days/hours etc. enabled us to quickly visualize the importance of hours in bike use. It's obviously logical when you think about it, but having the information already encoded saved us a lot of time.

Knowing that dates and time periods were going to be of paramount importance, we quickly sought to understand the dates of our X_train and X_test on Kaggle, as well as the specifics of the time periods in our dataset. We quickly realized that **our train dataset would run from 01/09/2020 to 09/09/2021** (our local test ran from 10/08/2021 to 09/09/2021) and that our **test on Kaggle would run from 10/09/2021 to 18/10/2021**.

b) Covid was an important parameter to take into account

Indeed, more than **two-third of our training set** portrays a period during which covid-related restrictions (confinement, curfew) affected bicycle use in Paris.



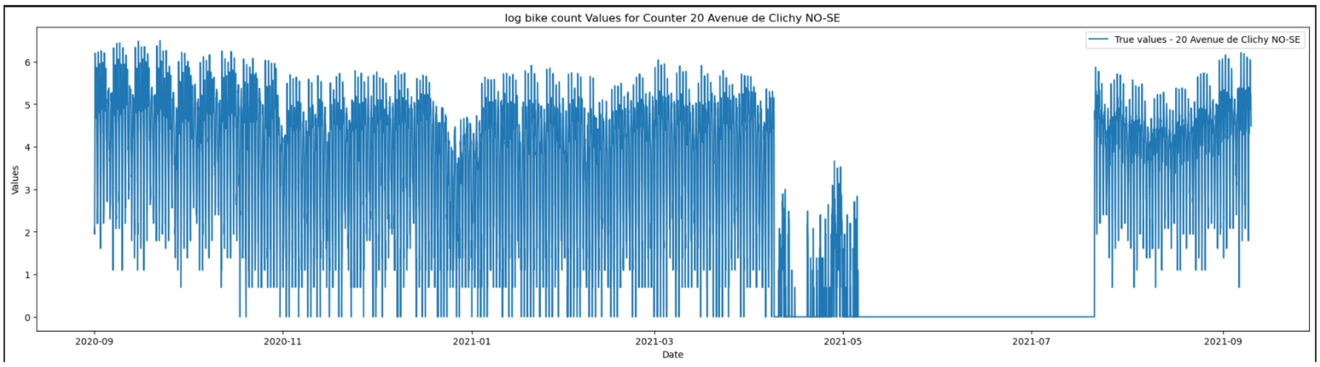
So we had to be careful, because it seemed highly likely that the distribution of data from the public train and the private test set would vary enormously, since **the test set is not influenced by covid**.

c) Which model to work on

Initially, we tried out various models on the data as is. Two models quickly stood out for their performance: ***CatBoost* and *XGBoost***. This came as no surprise, as these two models are particularly well known, especially on Kaggle competitions, for being **robust despite missing values and the mix between categorical and numerical data**, as tabular data can be. For a while, we tried to grasp typical Times Series Analysis models (ARIMA or SARIMA), but we realized that these were more useful for deducing predictive models to guess future behavior and not a Kaggle-type competition where the objective is to minimize a single value such as the RMSE. We therefore opted for the two *CatBoost* and *XGBoost* models, but **in hindsight, we felt that other, less "complex" models might have been a good idea (see in IV.b)**

d) Issue of the counters

During EDA, we noticed that some counters were missing data or had strange and especially low bike counts over a long period of time, for example in Montparnasse, Clichy and Pompidou counters. This was particularly the case for counters on the rue de Clichy in both directions (here in NO-SE direction):



We therefore tried to implement several techniques to weight/eliminate the effect of these zero values. In particular, we made the model run after deleting these values, replacing them with an average value of all the other values, we also tried implementing a binary variable to indicate periods when the counter seemed faulty, or replacing weeks where values were missing with "normal" weeks to limit the impact.

II - External data and feature engineering

a) Weather dataset

i) Preselecting variables

The weather data we had to start with had two major shortcomings: **a lack of temporal consistency with our original dataset**, the weather dataset having erratic times and often operating in three-hour increments, unlike our data, which operated hour by hour. **The second shortcoming was the sometimes staggering number of missing values in each column**. We read the weather data documentation available to us and made an initial selection of variables we felt were relevant: temperature, windspeed, rainfall and humidity.

After observing the data more closely and realizing that many of the data we had taken were of little relevance in training our model, via a drop in our local RMSE, we were finally obliged to remove a number of variables, some like humidity or wind speed, which seemed relevant to us at the outset. Others were removed because the information they contained was redundant, such as the minimum temperature in 24 hours, which was logically too closely correlated with the hour-by-hour temperature. **In the end, we decided to keep only 2 meteorological variables: precipitation and temperature**. We tried to find other, more exhaustive datasets, but for want of anything better, we concentrated on this one.

ii) Engineering the data

After numerous trials, which we won't go into here, **we finally opted for linear interpolation to fill in the missing values in the weather dataset**. Once all the values had been obtained, we tried to maximize the information contained in the new data. Leaving the numerical values as they were, probably because of their non-Gaussian distribution and too many over-represented outliers, didn't give such good scores as encoding them. We therefore opted for a basic encoding of the temperature in several large slices, as well as a binary encoding (with a precise rain rate chosen by us) to signify whether it was raining or not.

b) Covid and holidays data

We modeled Covid using the same encoding logic. We looked for date ranges concerning confinements and curfews and modeled their impact accordingly with: **(2) if the dates were subject to strict confinement, (1) if subject to curfews or period requiring attestation, (0) if no Covid restrictions**. Similarly, we found the dates concerning vacations over the period covered by our dataset and encoded them **(1) if vacation period and (0) if not**. Although this method may seem simplistic, it gave us our best results and we've kept it.

c) What we tried to add unsuccessfully

i) Adding external variables

We tried to represent daily and monthly cyclicity with sine and cosine functions to avoid the jump in values between midnight (coded 24) and 1am (coded 1) or between the 31st of the month and the 1st of the next. We also tried encoding public holidays and strike days, but to no avail.

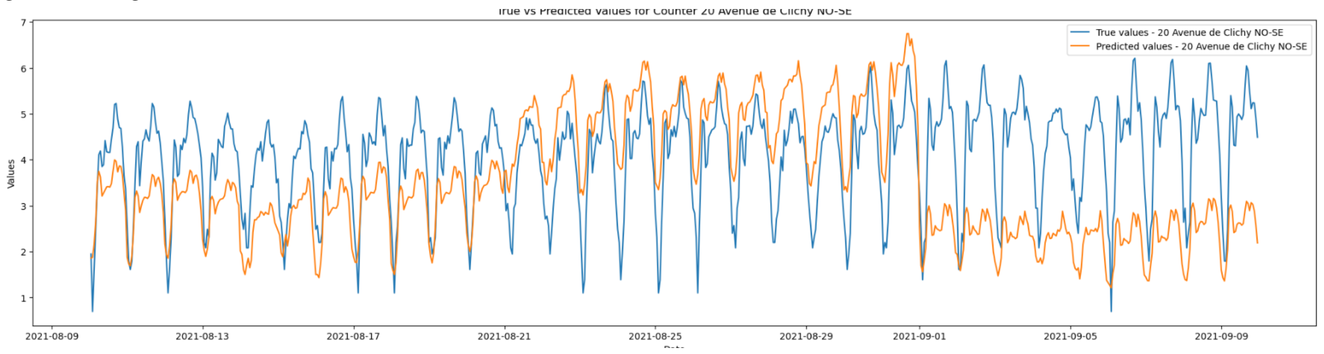
We tried to create columns based on correlations between values in the same column and previous values, in order to potentially identify a pattern that our estimator could recognize for predictions. With an autocorrelation function, we spotted whether patterns within rain and temperature were repeating themselves, when we still had the numerical values, and tried to build columns representing "lags/shifts" but this didn't improve our score.

Finally, as a good practice on Times Series, we tried to **remove a small percentage of the end data** from our X_train to avoid data leakage and to get a more realistic assessment of our model, but this was not convincing.

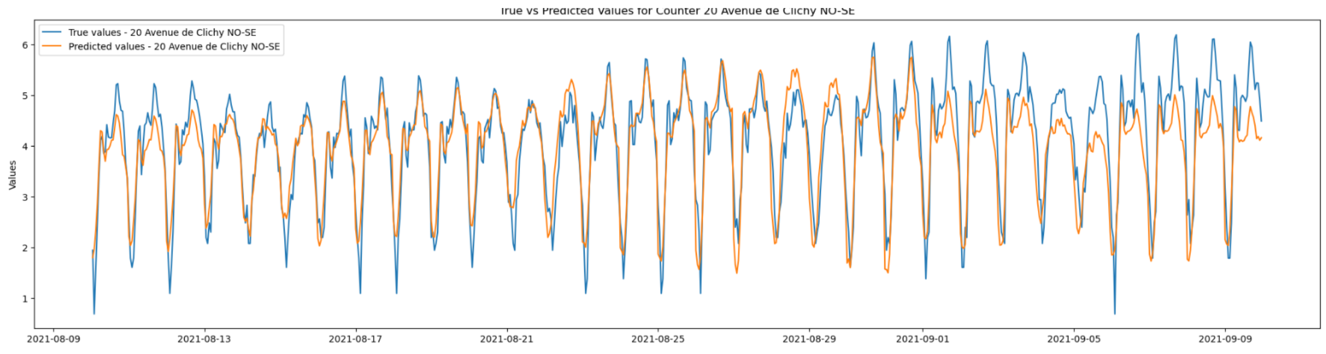
ii) Counters

As regards the counters we had identified with zero or missing values over a long period, our initial attempts to weight these periods (using a binary variable to identify periods when the counter seemed deficient, or deleting zero values, or replacing zero values with an overall average value) **were inefficient**.

We had a deeper analysis about the impact of the absence of values on our predictions. We calculated the RMSEs per counter and found that **there was a major problem on Avenue de Clichy** (all counters had **RMSEs between 0.3 and 0.7**, while those on avenue de Clichy had **RMSEs of over 1.4**). Here was the prediction against the y_test on the avenue de Clichy, and one can clearly see that the predictions were not good enough.



We therefore **replaced weeks with missing values by weeks with normal behavior**, thinking that this would be closer to the hypothetical reality, since replacing with an average, for example, would not show the difference in passage per hour. This proved to be the right decision, as our RMSE on the Clichy counters fell **below 0.7** on our test set, saving us around 5 hundredths of an RMSE on our local test. Here the predictions were way better:



Unfortunately, **this model didn't improve our RMSE on Kaggle**. We had submitted this model anyway, as we thought it might enable us to gain points on the private leaderboard, but this was not the case. Given that it had worked locally, we believe that **the Clichy counters may have had an anomaly during the Kaggle test period**, which we (and our model) were unfortunately unable to detect.

III - The model itself

a) *XGBoost* tuning

To tune the hyperparameters, we tried a lot of things. Overall, **the difficulty lay in cross-validation**. We knew it was preferable to do a *TimeSeriesSplit*, but this required sorting our data in chronological order, and for reasons we couldn't determine, **reordering our data significantly changed the accuracy of our models**.

So we simply applied **Grid Search** on the following parameters that we had identified as the most decisive by reading the literature: ***learning_rate, max_depth and n_estimators***, and then also ran ***Optuna*** on the X_test by looking at these hyperparameters. This enabled us to significantly increase the model's performance, notably by changing the n_estimators, which we increased on our final submissions. This represents the total number of decision trees to be trained in the model.

However, we also have to admit that the best results we were getting locally were not the best we were getting on Kaggle. So **we also tried tuning the hyperparameters little by little**, making several successive submissions, and this proved conclusive.

b) What we tried to tune unsuccessfully

One of our major problems was with **categorical variables**. Already with *Catboost*, where we had to clearly identify categorical variables by specifying them in the hyperparameters, this had caused us problems, as specifying certain categorical variables such as months or days increased our RMSE.

On *XGBoost*, the problem was similar. We had set the hyperparameter enable_categorical=True, but we had to set the categorical variables astype('categorical'), and once again, when we changed certain numerical variables to categorical (because they reflected something categorical), this increased the RMSE. So we simply kept in categorical the values that had improved our RMSE once switched to categorical, aware that this was certainly not very healthy, but it was more efficient.

One of our other difficulties was to identify our most important variables and clean up our dataframe of features that would overfitter it, or simply not add value to our model. To do this, we used two functions from the *XGBoost* library: .feature_importances and .plot_importances. One returns the weight of each feature, while the other returns the number of times each feature has been used to create a new branch of the decision tree. The first difficulty was that **some features seemed important with one function but not with the other**, so it was complicated to make a decision. Secondly, we tried to remove the variables that seemed the least important, and this deteriorated the performance of our model. In the end, **we decided to keep all our features**.

IV - Conclusions

a) What we managed to understand and implement

Although the majority of our initiatives did not lead to significant increases, **we did manage to identify guidelines that helped us improve our score**. We quickly realized that **many of the categorical variables** in the initial dataset **contained the same information and could be removed**. We deleted sites_name, counter_name, installation_date, technical_id and geographical coordinates, as we couldn't find any relevant way of using this information. We also realized that **the more simply we transmitted our external information** (covid, vacations etc), **the more we would avoid overfitting and the better our score would be**, hence our regular use of encoding.

We're relatively **satisfied with the score we achieved (top 25%)**, although **a little frustrated that we weren't able to properly model our findings on counter malfunctions**. This Kaggle being our first data challenge we were surprised at how difficult it was to find relevant and clean external data, and had to engineer it accordingly. Surprisingly, we ended up with a very small number of external parameters and a rather simplistic model, albeit quite effective in its predictions.

b) What we could have done better

In hindsight, we feel that perhaps we shouldn't have gone for the best Gradient Boosting Regressor models such as *XGBoost* and **instead focused on more simplistic models that were more understandable** and therefore less prone to misunderstandings when, after adding a significantly explanatory variable, we couldn't understand why our score was dropping accordingly. Similarly, we were unable to understand why our RMSE could fall locally and yet increase on Kaggle. The opacity of our models prevented us from understanding how the specificities of the X_test negatively influenced our trained estimator on the X_train. Conversely, **we sometimes fumbled too much in the hyperparameterization of our model** (cf. inability to maximize our GridSearch, our Optuna or our temporal cross-validation), which **explains our large number of submissions**.