RAMP PROJECT: PREDICTING CYCLIST TRAFFIC IN PARIS

Simon Mack
X-HEC
Paris, France

Adrian Tan
X-HEC
Paris, France

December 8, 2021

ABSTRACT

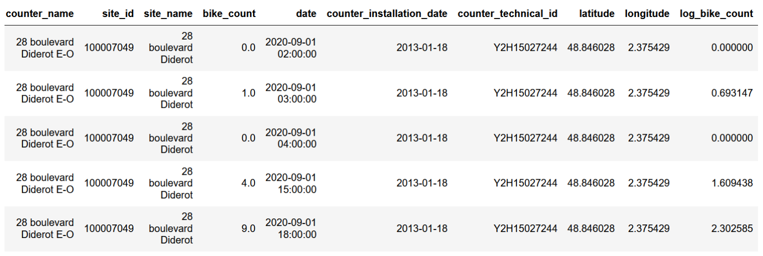
There are numerous bike counter machines located throughout Paris that record how many individual bikes pass by the specific counter on an hourly basis. The objective of our project was whether we could accurately predict daily traffic at each specific counter for the upcoming months and whether we could accurately model any rush hour traffic spikes. The performance of our final model was ultimately decided by an unknown data-set that we could not train or test on.

1 INTRODUCTION

As Paris is moving away from a car-centric model, it has been investing in expanding its bikelanes and making the city more friendly towards bikers. The bike counters, which measure the amount of bikers heading in both street directions, help keep track of bike traffic on specific Parisian roads. Determining the amount of a bikers at any given moment or location can be considered a challenging task. However, there are many variables that can help make a more accurate predication model. Our project therefore aims to develop a prediction model using any model of our choosing.

2 DATA AND ITS PREPROCESSING

The data source for this project is open source data provided by the city of Paris, consisting of countless bike counters. For our project, we will only consider 30 specific locations. The original data set for the project contained 455163 rows



counter_name	site_id	site_name	bike_count	date	counter_installation_date	counter_technical_id	latitude	longitude	log_bike_count
28 boulevard Diderot E-O	100007049	28 boulevard Diderot	0.0	2020-09-01 02:00:00	2013-01-18	Y2H15027244	48.846028	2.375429	0.000000
28 boulevard Diderot E-O	100007049	28 boulevard Diderot	1.0	2020-09-01 03:00:00	2013-01-18	Y2H15027244	48.846028	2.375429	0.693147
28 boulevard Diderot E-O	100007049	28 boulevard Diderot	0.0	2020-09-01 04:00:00	2013-01-18	Y2H15027244	48.846028	2.375429	0.000000
28 boulevard Diderot E-O	100007049	28 boulevard Diderot	4.0	2020-09-01 15:00:00	2013-01-18	Y2H15027244	48.846028	2.375429	1.609438
28 boulevard Diderot E-O	100007049	28 boulevard Diderot	9.0	2020-09-01 18:00:00	2013-01-18	Y2H15027244	48.846028	2.375429	2.302585

Figure 1: First 5 rows of the original data set.

and 11 different columns. It is immediately clear that not all columns would help in constructing our prediction model; meaning that we were eventually left with the following variables to build our model on:

- site_name.
- counter_name
- date

The variables bike_count and log_bike_count were the variables that we had to predict, and therefore could not be considered part of our prediction model.

3 INITIAL DATA ANALYSIS

To get a better understanding of the provided data, we visualized the data with respect to a couple different time values and bike counters.

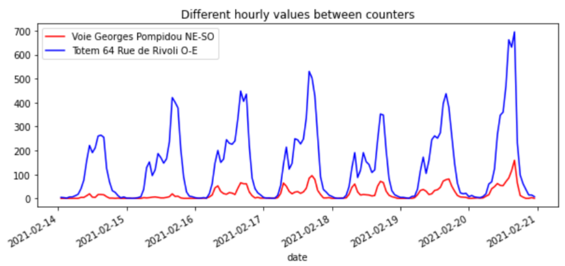


Figure 2: Hourly values

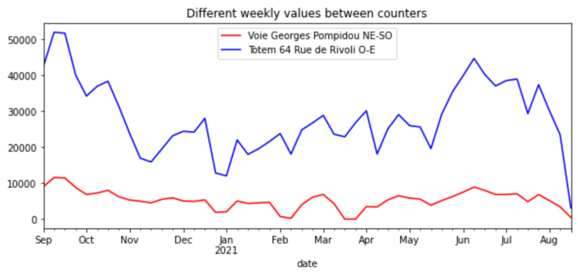


Figure 3: Weekly values

From these two graphs we can immediately see that there is a large discrepancy between different bike counters but also between different timestamps. This confirms our assumption that we must consider both the different bike counters in our model, as well as the timestamps. While these initial variables will get us quite far for our prediction model, it is likely that further external data must be utilized to help explain the variation in bike counts across different weeks and hours.

4 MODEL CONSTRUCTION

4.1 Initial Data Modeling

To get started on our model selection, we began by creating a model using solely the data provided in the original data set. The rationale behind taking this approach first was that we believed it helped us understand the limitations of the data in the original data set and would set a baseline on which the external data would have to improve on. The "strength" of a model would be determined by the RMSE value obtained. The lower the RMSE, the "stronger" the model.

4.1.1 Model selection

To begin, we tested out some of the more popular prediction models, as well as models that are claimed[1] to be especially strong for time series forecasting. The models that we tested were: (1) Linear regression, (2) Random forest regression, (3) Histogram-based boosting regression, (4) Extreme gradient boosting, and (5) Light gradient boosting method. Since

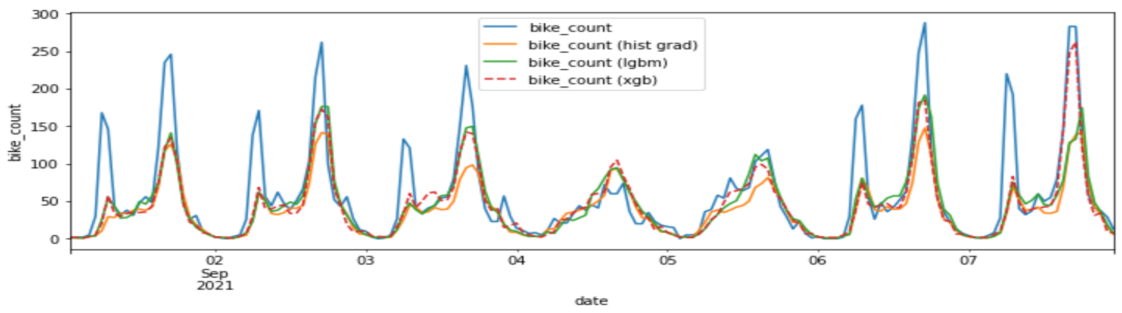


Figure 4: Comparing different predictive models

we are trying to map an output that has a continuous (real) value, we only need to consider regression functions. Upon running a couple simple tests with the previously listed models, we decided to continue with the HistGrad, XGBoost, and LGB models. Our rationale behind dropping the linear and random forest regressors was that they had the lowest RMSE.

4.1.2 Time-related feature engineering

Trigonometric dates
Once we selected the optimal model to predict our data, we attempted to decrease the RMSE of the model by redefining the dates of the data set. For the initial model selection, the date column was originally broken up into the following columns:

- year $\Leftrightarrow (2020, 2021)$
- month $\Leftrightarrow (1, ..., 12)$
- weekday $\Leftrightarrow (0, ..., 6)$
- day $\Leftrightarrow (1, ..., 31)$
- hour $\Leftrightarrow (0, ..., 23)$

While ordinally encoding the time in such a manner helps differentiate between different days of the week and hours of the day, the regressor function isn't aware of the fact that all of the variables are cyclical (e.g. weekday 6 is followed by weekday 0, hour 23 is followed by hour 0). The model would consider hour 0 and hour 23 as 23 hours apart, whereas they are in fact only 1. To combat this issue, we decided to cyclically encode all the time variables. By converting all our time variables as cosine and sine functions, we provide the machine with the hint that the variables are cyclical. We use both cosine and sine functions because there are duplicate values on a 2π sinusoidal wave and adding both functions prevents this from becoming an issue with the model.

We did not end up converting all time variables to a cyclical equivalent, however. After conducting several tests to find the most stable model and strongest RMSE, we determined that the day and year values would remain integers while the weekday, month, and hour variables would become cyclically encoded.

The fact that cyclically encoding the day variable had an adverse effect on our model was surprising, but it can likely be explained by the fact that there are little patterns to be observed by the day variables (e.g. days can fall on any weekday and are influenced by the month). Following this change, we recorded an improvement on our RMSE from **0.805** to **0.757**, $6\% \uparrow$.

Periodic spline features
After implementing the cyclical dates in our model, we discovered that there was an even stronger approach to encoding the dates and decreasing the estimation error of our model. Having already established that the day and year did not have to be encoded, we only transformed the weekday, month, and hour. The spline transformation is similar to that of the sin and cosine transformation, however the spline transformation results in a larger number of expanded features. The increased numbers of features helped the model differentiate between the different hour values while maintaining the cyclical characteristics of the data inputs.

Pairwise interactions
Finally, based on the visual interpretation of the data we can see that there is an interaction between the weekday value and the specific hour of the day. To include this interaction within our model we created a pairwise interaction within our estimator. More specifically, we created a new variable that differentiated between a 'workday' and the weekend and subsequently made a separate pipeline that solely modeled the interaction between the hour of the day and whether it was a workday. This separate pipeline was then included in the final pipeline using the *FeatureUnion* function.

4.2 Leveraging external data

After cleaning and improving the original data-set, we moved on to finding external data that could help explain certain spikes and troughs that we observed in the data visuals. The original data set came accompanied with an external CSV file containing various weather variables.

Having analyzed our initial data visualization, it was immediately clear that there were external factors contributing to the spikes and decreases in bike count during certain hours, days, weeks, and months. To help explain these anomalies, we decided to try out the following external variables for our model:

- Rush hour: $Binary : x_i = \{0, 1\}$
- Curfew: $Binary : x_i = \{0, 1\}$
- Holidays: $Binary : x_i = \{0, 1\}$
- Google Transit Data: $Discrete : x_i = [0, \infty]$
- Brent crude price: $Continuous : x_i = [0, \infty]$
- Car traffic: $Discrete : x_i = [0, \infty]$
- Temperature: $Continuous : x_i = [0, \infty]$
- Wind speed: $Continuous : x_i = [0, \infty]$
- Humidity: $Continuous : x_i = [0, \infty]$
- Google Workplace Data $Discrete : x_i = [0, \infty]$

To determine whether they had any influence on our model, we graphed the feature importance of the 3 different regressors to determine which variables to keep.

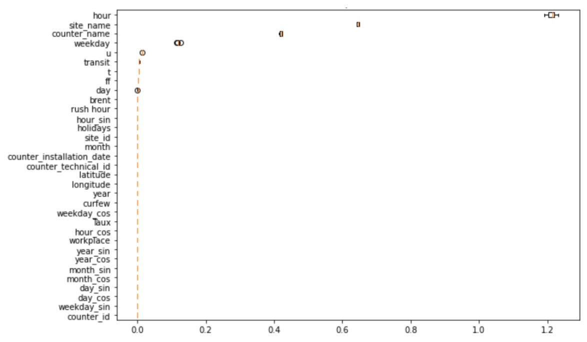


Figure 5: Feature importance of data set variables

All three different models allot the same importance to all the different variables in the external data and in the original data set. Based on this information, we ultimately decided to select the variables highlighted in yellow (from the previous list) to include in our regressor.

4.3 Hyper parameter tuning

After determining the external variables we were going to use and confirming that they had a positive effect on the RMSE, we started adjusting the hyper parameters of our model. To find the best parameters, we used the TimeSeriesCV package to help in selecting the optimal criteria. It was decided to use TimeSeriesCV instead of GridSearchCV because the former did not use future data (i.e. data points whose datetime would be after the test set). This seemed a more appropriate approach to finding the right parameters and prevented the CV from using future data to predict past data, as a GridSearchCV might have done. We tested the following parameters for our regressors:

HistGrad	XGB	LGBM
random_state	random_state	random_state
max_leaf_nodes	num_leaves	max_depth
max_iter	n_estimators	random_state

For any of the random_state in our regressors, we kept the value = 0 to ensure that our results would be replicable across different attempts.

5 Results

After adjusting our models and trying to fit them with the best parameters, we graphed the final estimations to get a better visual understanding of how our models had improved and where it was still lacking. It was, however, hard to

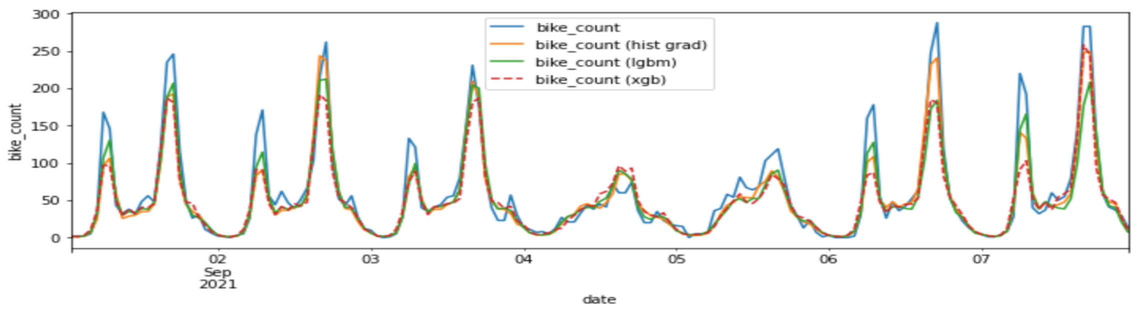


Figure 6: Final estimator results for a given week

decide what the best estimator was based solely on our visual data. We ran numerous local ramp tests to determine which model had the lowest RMSE score for both the validation sample and the test sample. We ultimately decided to use the **LGB model** as our submission; it had the strongest RMSE across the local test environment and the studio environment and also had the fastest train and validation times.

6 Conclusion

To conclude, it has become apparent that the optimization of the model is dependent on several different variables: the model selected, the data selected, and the manner in which the data is presented to the model. Further improvements that might be studied for the project would include additional data cleaning (e.g. how to best replace NaN values) and leveraging more complex external datasets.

7 Acknowledgement

We would like to thank Roman Yurchak for his guidance throughout the project and for creating this challenge. Furthermore, we would like to thank Mathurin Massias and Hicham Janati for their support throughout our Python course and for their helpful feedback.

References

[1] Reganuth Gavita. 10 incredibly useful time series forecasting algorithms. *Advancing Analytics*, 2021.