Kaggle challenge

Bike traffic counters installed by the City of the Paris

Adrien Senghor, *X-HEC DSB | HEC Paris Student*
Victor Soto, *X-HEC DSB | HEC Paris Student*
Github Link

Introduction

The Kaggle challenge was about **predicting the log hourly traffic at each bike counter in Paris** for a given period (2020-09-01 01:00 to 2021-09-09 23:00). The efficiency of predictions was evaluated on the **root-mean-square error (RMSE)** between the predicted log-count and the evaluated target. A training set with information on Paris bike counters between 2020 and 2021, along with external data about Paris weather conditions during the same period, was provided.

The study begins with an **exploratory data analysis** part (A) on the training set and on the external data. The features of the training set are very clean without any missing values. However, the labels of the training set (log-hourly traffic for each bike counter) are distorted for 3 sites because of measurement errors. External data has way more missing values (up to 90% for some features) and the missing values were filled with the last non missing value or the most frequent modality depending on the features. Some date features were built (notably dates of holidays and dates of lockdown in Paris), but no other external source of data was used.

Having merged all the data in one dataset, **feature engineering** was applied to some features (B). Cosine transformation was applied to cyclical datetime features to modelize proximity between periods of time. Having only numerical features, a StandardScaler was applied in the pipeline to standardize and scale the selected features.

Choosing the right model (C) was done by iteration. Simple regression models were used at the beginning to compare which one would be the most efficient (model selection according to the RMSE computed on the test set). In the end, HistGradientBoosting regressor was the most suitable estimator. The selected best hyperparameters were selected using Optuna.

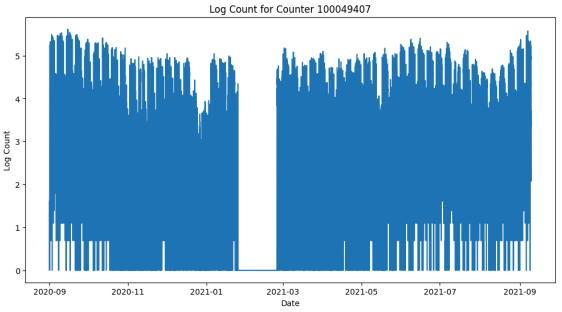
Ultimately, **the best prediction obtains a 0.62 score** on the Kaggle private leaderboard.

A. Exploratory Data Analysis (EDA)

1. Training Data

The first EDA focused on the training set features. It is worth noting that this set is clean, without any missing values except for 8.3% missing values in the y labels (bike_count & log_bike_count). However, features describing the location of the counters are highly correlated (site_id, counter_name, address, coordinates), some needed to be dropped to prevent from overfitting the model.

Regarding the 8.3% of missing y labels, a further analysis shed light on three bike counters that were likely broken for at least one month each, which explains the null outputs. Due to lack of time, the missing values were not filled. However, it can be assumed that those broken counters explain a part of the final error.



2. External Data

The first step dealing with external data was to understand the meaning of the 59 columns. After research on the internet, it was assumed that the external data had the same format as the open source weather dataset from the French government[1].

Another valuable insight about the external dataset is that all columns were not relevant for training the model. Indeed, some features were completely empty, while some others had between 45% and 90% of missing values. For the consistency of the training, it was chosen to keep only features with a maximum of 12% missing values.

B. Feature Engineering

1. Training Data

A big part of the feature engineering on the training set was about creating valuable insights from the X["date"] column. Based on prior suppositions, periods of time fostering that discouraged the use of bikes in Paris were created. Thus, X['is_weekend'], X['is_holiday'][2], X['is_covid'] were built from the X['date'] column.

[1] METEO - Observation_Station_MDM (SYNOP)

[2] Public holidays dates, *French government*, and Covid dates, *Le Monde*

Sine/cosine transformation was applied to datetime features. As described in the Nvidia developer blog[3], this cyclical transformation helps modeling the proximity between dates that are considered as far with a numerical approach (e.g model the proximity between the month of December (12) and January (1) with a polar representation of the months by applying a sinus and cosinus transformations to the initial information). Cyclical “season”, “month”, “weekday” and “hour” features were derived from X[“date”] with this approach.

2. External Data

The main issue with external data was about the feature engineering on external data was about filling the missing values that are not handled by linear regression such as the Ridge regression suggested. It was chosen to sort the external data based on the same chronology as the training data, and to fill the missing values with the last non missing value available. In the case this information was available, it was filled with the most frequent modality of the column. This strategy was chosen instead of filling with the mean value, because it made sense for weather data to be close to the last value. It ensured that seasonal tendencies were respected.

3. Data encoding

The last part about feature engineering was about encoding the data in the preprocessing of the pipeline. Having only numerical values, it was chosen to use a StandardScaler[4] module from scikitlearn. It was considered whether to use the OrdinalEncoder for the binary features (e.g X[‘is_covid’]), but since those features were already integers, and that the score wasn’t affected by this encoding (measured on a few iterations), it was decided not to add the binary features into the StandardScaler.

C. Model and hyperparameters selection

1. Feature Selection

Feature selection was conducted in two stages. In a first ‘naive’ approach, the selected features were the ones that seemed most relevant to explain the problem, while trying to avoid redundancy of the features. Therefore, redundant features from the training set were dropped (only the site_id information was used for the location, because it was comfortable to use a numerical feature).

In a second part, an iterative approach was made to test whether some features had an impact on the final RMSE. The external features that were the most correlated to the y label were spotted using a correlation matrix, shedding light on features for which the influence on

[3] Three approaches to encoding time informations as features for ML models, *Nvidia developer blog*
[4] StandardScaler module from scikitlearn is used to normalize and scale the values from a column

the y label could have been intuited (temperature, rain, wind, road conditions). By iteration, features were added one by one using the suggested Ridge regression on which a simple pipeline had been applied; a final set of external features was then selected arbitrarily, based on the precision measures derived from the iterations. This approach was not the most rigorous, but helped considering the external data in what appeared to be the best way within the time constraint.

2. Model selection

Various regression estimators were tried with their default parameters, so that the best model could be selected based on the final RMSE on the test set (using a 5 fold cross-validation for more confidence). Here are the results, using external data and with the developed Feature Engineering applied:

Estimator	Ridge*	RandomForest	CatBoost	HistGradBoost
Ridge (without external data)	0.90	0.98	0.71	0.72

**score obtained without any external data*

3. Hyperparameters finetuning

Because of lack of time and understanding of the CatBoost model, also considering that the HistGradientBoosting[5] model is optimized for large datasets with numerical features, the latter was chosen for the final submission. Optuna[5] was then applied to finetune the following hyperparameters: learning_rate, max_iter and max_depth.

Conclusion

The Kaggle challenge on predicting the bike traffic in Paris provided valuable insights into data preprocessing, feature engineering, and model selection under practical constraints. Despite challenges such as missing data in external datasets and distorted labels for some bike counters, systematic approaches to cleaning, encoding, and feature selection proved effective in building a predictive model.

The use of cyclical transformations for datetime features and the integration of contextual data (holidays, weekends, lockdown periods) proved to be valuable. The adoption of HistGradientBoosting Regressor as the final model, optimized using Optuna, can be considered as a sound choice given the dataset's characteristics. Although external data was leveraged selectively to mitigate the impact of missing values, a more thorough exploration of feature relevance might have improved the final result.

[5] Optuna is a hyperparameter optimization framework