Predicting Cyclist Traffic in Paris

Ameya Kulkarni Qiling Zhu

December 2023

Outline

1. Exploratory Data Analysis
2. Feature Engineering
3. Model Exploration
4. Possible Improvements

1 Exploratory Data Analysis

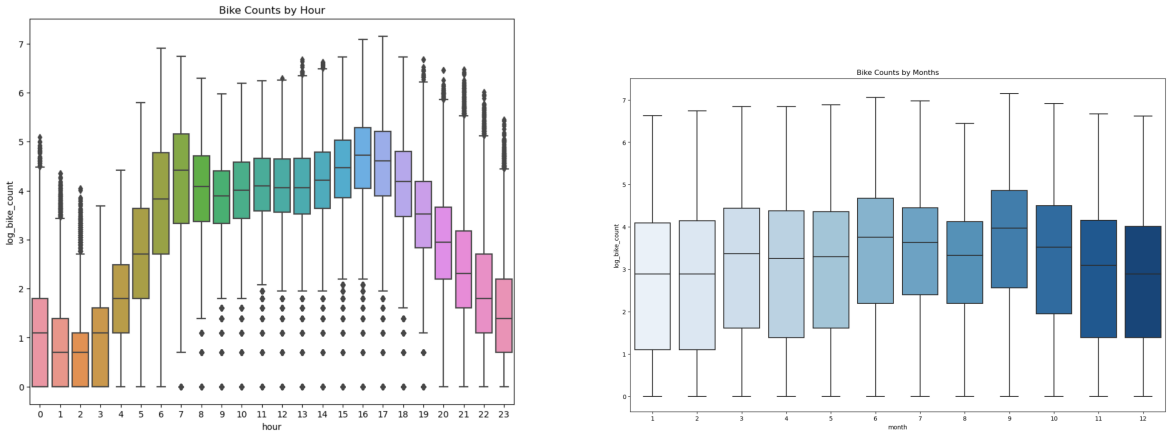
1.1 Data Set Exploration

We first examined the general information of the *train.parquet* dataset. The dataset comprises 455,163 observations and 11 features, with no missing values found. Two columns serve as the dependent variables: ***bike_count*** and ***log_bike_count*** (we focus on the latter). The remaining 10 columns represent explanatory variables, including names/id features (***counter_name, counter_(technical)_id, site_id, site_name*** with dtype: category), geographical data (***longitude, latitude*** with dtype: float), and datetime features (***date, counter_installation_date*** with dtype: datetime).

Additional external data include weather-related metrics, matched with the corresponding dates of recording.

1.2 Time Series Trends of Log Bike Count

The analysis proceeded to uncover temporal patterns in bike usage by examining (1) hourly and (2) monthly log bike counts.



(a) Hourly distribution of log bike counts

(b) Monthly variation in log bike counts

Figure 1: Time series trends of log bike counts by hour and month.

Analysis of hourly log bike counts (1a) reveals significant fluctuations throughout the day, peaking between 6 a.m. and 6 p.m., notably around commuting hours (6 a.m. to 8 a.m. and 4 p.m. to 6 p.m.). This pattern aligns with typical work and school schedules.
Monthly trends (1b) shows no significant difference throughout the year, although a modest decline observed during winter can be observed, which seems plausible.

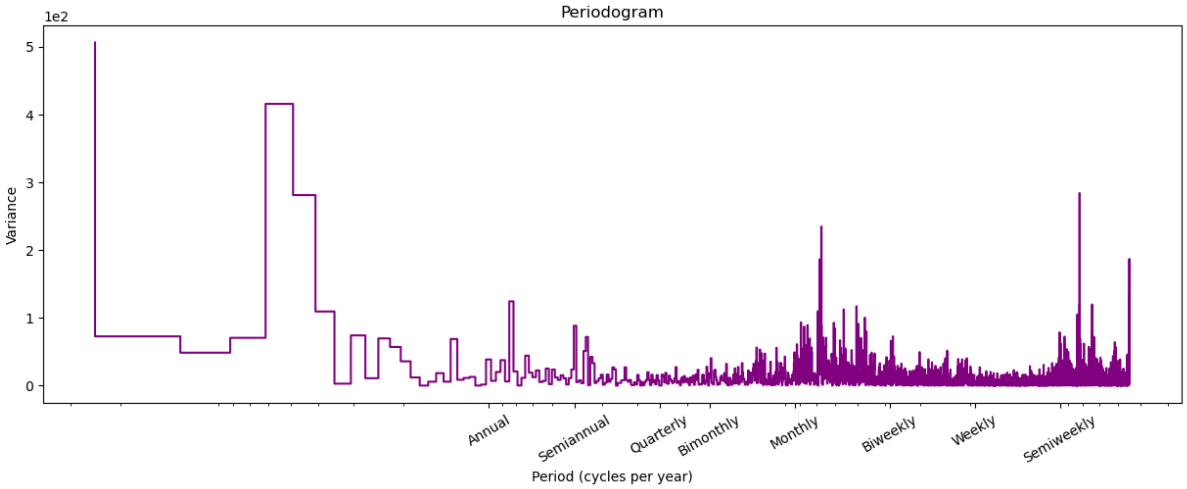


Figure 2: Periodogram illustrating the distribution of variance in bike counts across frequencies

2 Feature Engineering

To avoid redundancy and unnecessary information we removed less informative name/id-related features, the ***counter_installation_date*** column, and the ***bike_count*** column in the train dataset, and kept ***counter_name***, ***date***, and ***log_bike_count***.

2.1 Time Series Data

The Periodogram shown in Figure 2 indicates the variance of bike counts across frequencies. The peaks suggest regular cycles, with significant annual, weekly, semiweekly, and daily variations.
To leverage the temporal data, we engineered features from the ***date*** column, creating ***hour, dayofweek, month, quarter***, and ***dayofyear***. A binary ***is_weekend*** feature signifies weekends, and sine/cosine transformations of ***hour*** and ***dayofweek*** capture cyclical patterns. The ***date*** column was subsequently dropped.
Holidays may significantly influence bike usage, we therefor add a binary ***is_holiday*** column to indicate whether the day is a French holiday. Our data period covers the year 2020 and 2021, when the COVID-19 pandemic occured. Thus, we also included binary indicators for various lockdown periods.

2.2 Weather Data

Columns with excessive missing data or low relevance were omitted. The remaining variables underwent NaN value imputation, with categorical types replaced by the mode and continuous types by the mean, based on daily or weekly data. Anomalies such as negative values were corrected. The dataset was resampled to align with the train dataset's hourly format, with lag features introduced for continuous variables.
To mitigate collinearity, we compute the correlation coefficient matrix and pruned the features with absolute value higher than 0.8.

2.3 Interaction between Time and Weather Data

Investigating the weather dataset, we integrated interaction terms, such as one between temperature and hour to examine their combined effect on cycling patterns.

2.4 Additional Data

Accessibility to bicycles is another potential influencer. We quantified the proximity of Velib stations (a public bicycle sharing system in Paris) to each counter, calculating the ***distance_to_nearest_velib*** via the nearest-neighbor search with cKDTree.

2.5 Feature Importance

Overfitting is a concern when too many features are present. Figure 2 illustrates this risk, showing that an excess of features can elevate the RMSE. Utilizing XGBoost's feature importance metric, we iteratively removed less significant features to refine the test RMSE.

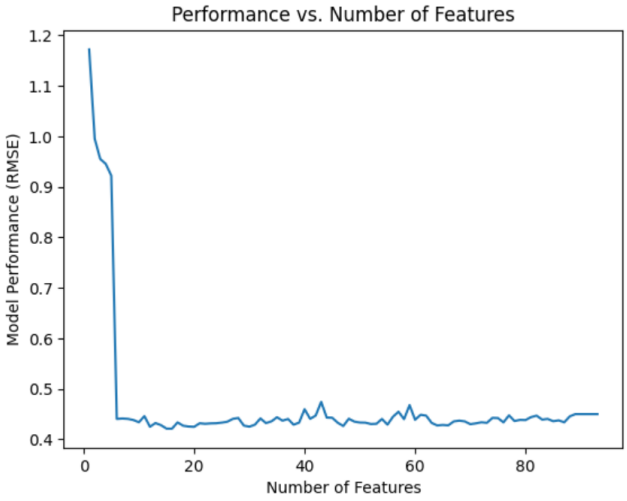


Figure 3: Number of features used and model performance

3 Model Exploration

3.1 Model Trials

Multiple models and gradient boosting frameworks were tested. We mainly focus on their performance in handling categorical data and capturing the complex relationships within the time series data.

SGD To build a baseline model, we followed the scikit-learn guidelines and chose stochastic gradient descent regression. We got an RMSE of 0.867 on the leaderboard.

CatBoost CatBoost is known for its gradient boosting framework designed to handle categorical variables. Since the task is to predict the bike counts for different counters (a categorical variable), this method is considered.

LightGBM LightGBM uses a leaf-wise growth strategy, which can be more effective in capturing complex relationships in the data. It also efficiently handles categorical features than XGBoost without one-hot encoding.

XGBoost XGBoost is commonly used for time series prediction, as it excels in handling non-linear patterns, provides regularization to prevent overfitting. It also shows efficiency in parallel computing (prediction for multiple counters).

Other Models We also tried other models and methods, such as **Adaboost** and timeseries models like **Prophet**, but they were eventually eliminated either due to model complexity or due to subpar performance.

3.2 Model Comparison

Initial models were trained exclusively on time series features, with the least influential features pruned to observe the change in RMSE. This process was replicated inclusive of weather-related data and its corresponding lag features. Table 1 below shows the performances of each method with different choice

of features. We can see that (1) model shows lower RMSE after dropping least important features, (2) model using XGBoost outperforms CatBoost.

Features Selected	Model	Test RMSE
All TS features	XGBoost	0.4284
After dropping	XGBoost	0.4234
	CatBoost	0.4227
All TS + weather	XGBoost	0.4453
	CatBoost	0.5073
After dropping	XGBoost	0.4440
	CatBoost	0.4722

Table 1: Model performance comparison based on feature selection

We used recursive feature elimination (RFECV) with time series cross validation to select our best features for model training, and performed hyperparameter tuning with Optuna to get the best hyperparameters across all the splits. Table 2 below shows the performances of each method using final selection of features. Among all methods XGBoost provided the best result.

Method	Test RMSE
CatBoost	0.4571
LightGBM	0.4810
XGBoost	0.3996

Table 2: Performance metrics post-feature reduction

3.3 Final Model

We chose to submit the model with the tuned XGBRegressor, which gives the best performance on the final test set. For feature selection, we saw that adding our weather data actually makes the model performance slightly worse, hence our best submission contains only the time series features data (apart from counter name), and few interaction terms between time and weather attributes achieved a RMSE of 0.6464 on the private leaderboard.

4 Possible Improvements

Feature Engineering The significant impact of temperature on model performance suggests a reevaluation of other weather features. Eliminating or re-engineering those with minimal importance may reduce noise and enhance the predictive power of the model. Further experimentation on developing features that better represent the interaction between time and weather could be beneficial.

Temporal Dynamics Analysis Deeper analysis into the model's handling of seasonal and temporal dynamics is suggested. An in-depth understanding of the influence exerted by features like ***dayofyear*** and ***month*** could inform more nuanced feature engineering.

Model Complexity The model's reliance on a few key features, notably ***hour_cos***, raises questions about the optimal complexity required. A simpler model may reduce the risk of overfitting, especially when we have numerous features with low importance.Moreover, Integrating other models/methods could lead to more accurate predictions: such as incorporating time-based predictions from Prophet into the XGBoost model, or adapting Adaboost's methodology (despite the current incompatibility with our XGBoost estimator, it remains a potential avenue for exploration)

Additional External Data Incorporating external factors such as demographic statistics, economic indicators, public events, and infrastructure developments could provide a more holistic view of the factors influencing bike usage patterns.