Project Report: Predicting Cyclist Traffic in Paris

Roy Bensimon & Raphaël Guenoune

Msc Data Science & AI For Business X-HEC - Ecole Polytechnique - December 2024

1 Data preparation

a) Original Dataset

To start, we carefully analyzed the dataset provided. The training set consists of 496,827 entries across 12 columns. This data was collected during a statistical study spanning from 01/09/2020 to 09/08/2021, recording the hourly count of bicycles at 30 different locations across Paris. Notably, the dataset is complete, with no missing values. We analyzed the distributions of bike_count and its log-transformed counterpart, log_bike_count. The log transformation significantly reduced skewness, improving data compactness and stabilizing variance. As shown in Figure 1, comparing the distributions of bike_count and log_bike_count, the transformation resulted in a more centralized mode. However, some asymmetry persisted, indicating that there is still room for further normalization.

For encoding, we tailored our strategies to suit the characteristics of each feature. To handle temporal variables such as months, days, and hours, we opted for cyclical encoding, as these features exhibit recurring patterns. This approach allows the model to recognize the periodic nature of time and its influence on bike usage. For instance, as shown in Figure 2, the log of bike counts follows a clear cyclical pattern throughout the day. However, one drawback of cyclical encoding is its compatibility with decision tree-based algorithms. These algorithms process features individually, which can lead to difficulties when working with the sine and cosine values of cyclical encoding in the same split. For the counter names, we chose one-hot encoding, as it effectively handles categorical variables without implying any hierarchical structure. Additionally, we standardized all numerical features using the StandardScaler, which significantly boosted the performance of our model. To capture seasonal trends, we created an interaction feature combining the month and hour, as bike usage often varies significantly depending on the time of day and the time of year.

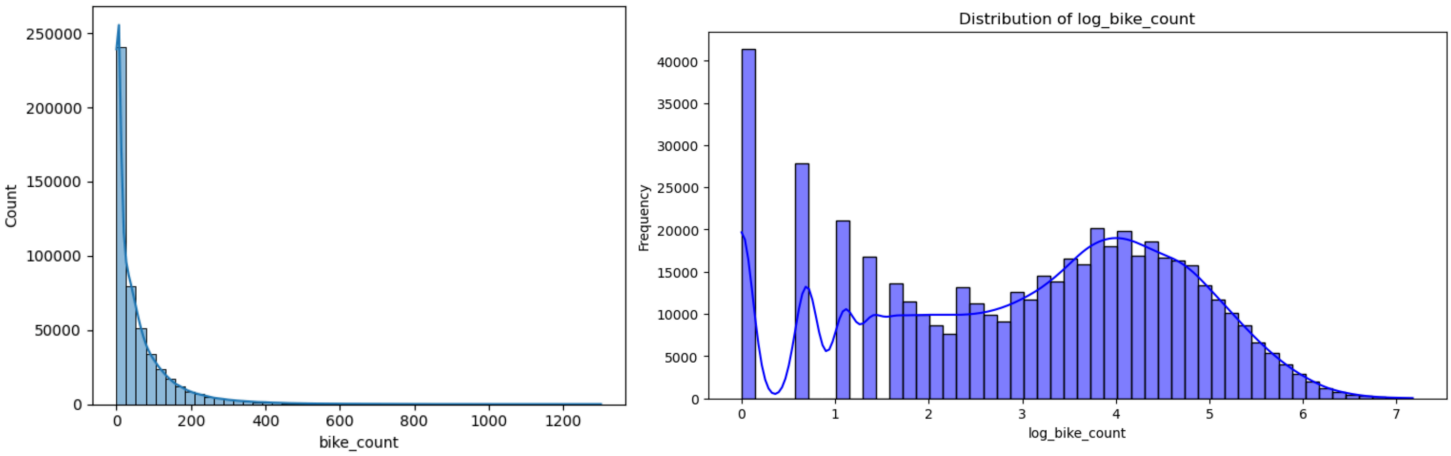


Figure 1: Comparison of Bike Count and Log-Transformed Bike Count Distributions

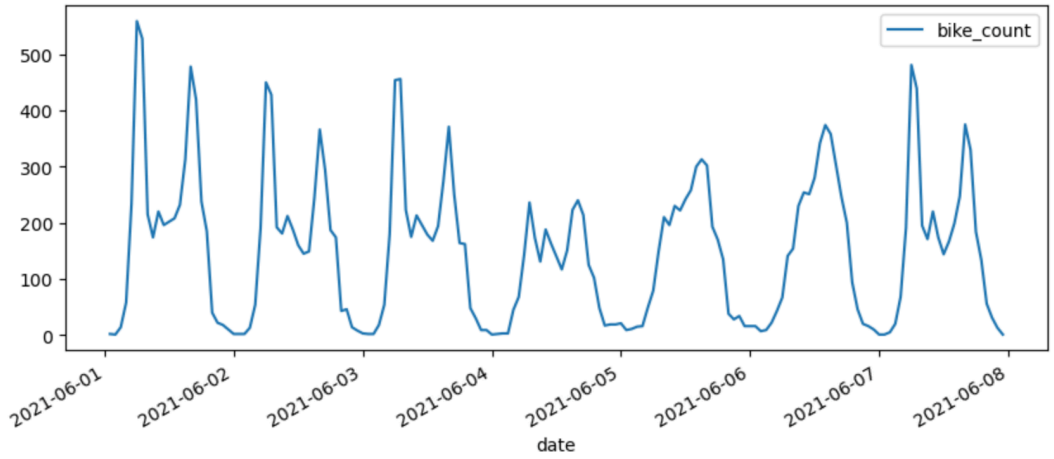


Figure 2: Variation in bike counts over the hours of a week across all counters.

b) External Dataset

We incorporated a weather dataset from the Paris-Orly weather station, containing 59 numerical features recorded every three hours from September 2020 to October 2021, aligning with our original dataset. To reduce the risk of overfitting, we addressed dimensionality and missing data by removing features with over 10% missing values and filling the rest using the median to minimize sensitivity to outliers. Additionally, we added temporal features such as boolean indicators for school holidays, bank holidays, and lockdown periods, as these factors influence bike usage. For example, lockdowns significantly reduced bike counts due to restricted outdoor travel. Given the sharp decline in bike counts during the Christmas holidays, as shown in Figure 3, we included a specific feature for this period to enhance the dataset.

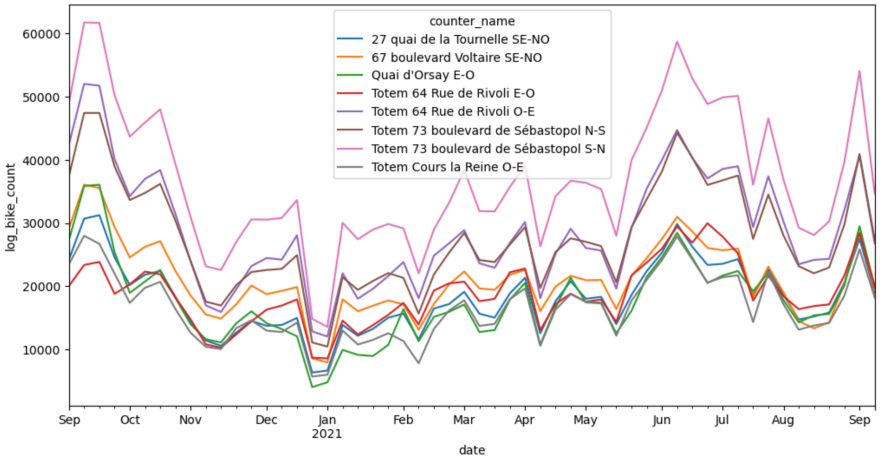


Figure 3: Weekly bike counts plotted over time, emphasizing the decline during the Christmas holidays

To integrate the external dataset with the original one, we used the `merge_asof` function from the pandas library. This method performs a left-join, matching rows based on the closest key rather than requiring an exact match. In our case, the key was the date, and to ensure the merging process worked correctly, both DataFrames had to be sorted beforehand by this key. This approach allowed us to align the datasets efficiently while preserving the temporal structure of the data.

2 Exploratory Data Analysis

Once the dataset was prepared, we explored the relationships between the dependent variable (log bike count) and the predictors to evaluate the relevance of features. We analysed a variety of

visualizations. The figure below highlights the valuable insights gained from this approach, helping us determine which features were most relevant for inclusion in our model. Several variables from the external dataset exhibited promising relationships, although not necessarily linear, with the log of bike counts, notably "christmas," "u," and "t." Furthermore, we observed that many variables lacked a linear relationship with the output (e.g., pmer, tend, dd, ff, vv), making models like Ridge Regression unsuitable and therefore best avoided.

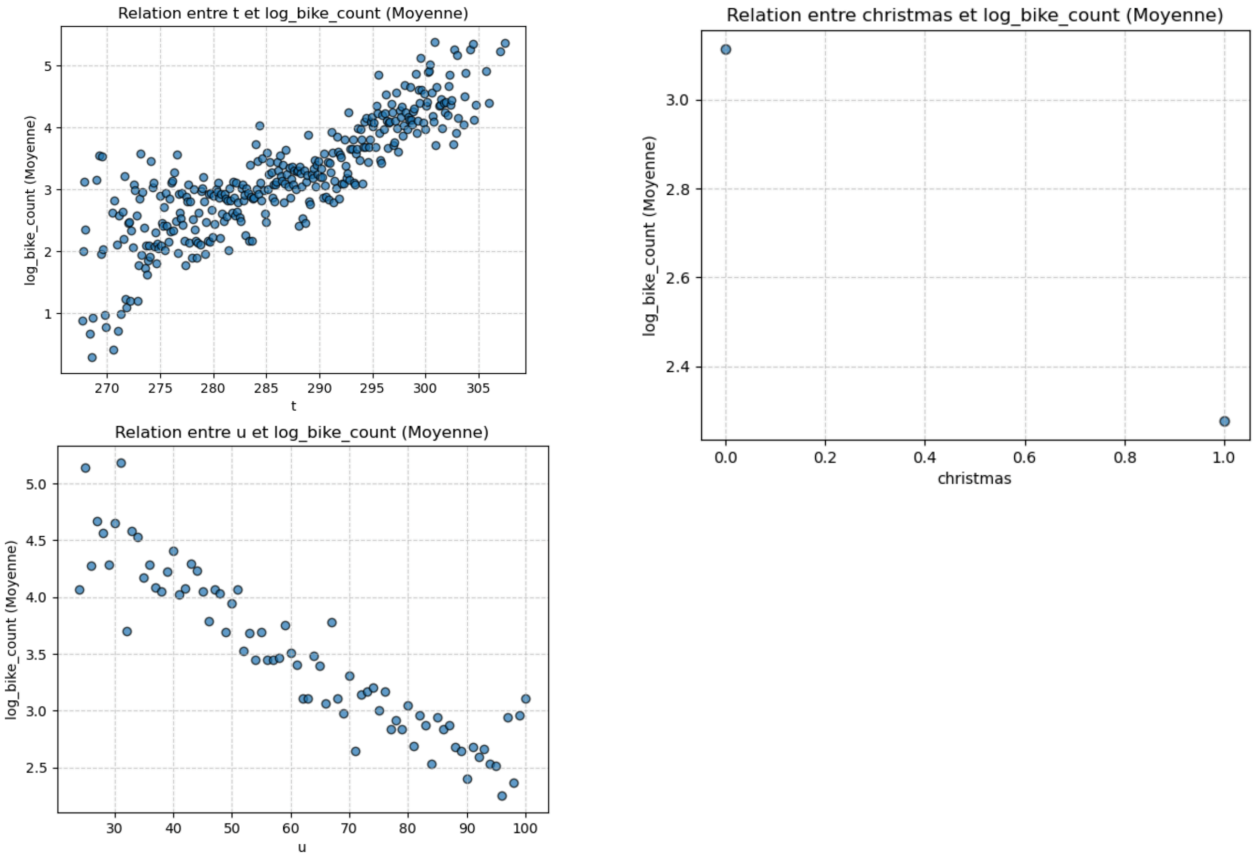
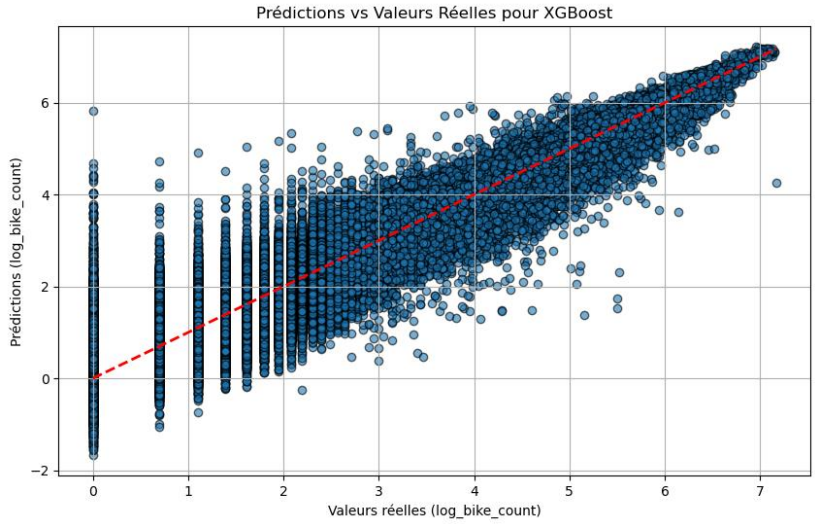


Figure 3: Linear relationship between external features and the target variable

3 Model Selection and Conclusion

Tree-based algorithms, particularly Random Forest, XGBoost, quickly outperformed linear regression. This outcome was anticipated, given their ability to handle non-linearity and collinearity far better than linear models. Additionally, these algorithms incorporate mechanisms to reduce overfitting, making them robust for this type of problem. Among the tree-based models, boosting algorithms demonstrated superior performance. Their iterative approach to minimizing errors, such as the RMSE in our case, significantly improved their predictive accuracy. After evaluating theses models, XGBoost emerged as the top-performing model. To refine the model, we experimented with various external features, including and excluding them to assess their impact on performance. The best results were obtained with a carefully selected subset of features. Hyperparameter tuning was conducted using GridSearchCV, with a particular focus on optimizing the XGBoost model, given its superior results. We thus obtained the following hyperparameters: colsample_bytree: 1.0, learning_rate: 0.3, max_depth: 6, n_estimators: 200, subsample: 0.8. Finally, based on our submission on the Kaggle (before tuning), the best model (The one with the smallest RMSE is XGtboost with a RMSE = 0.6921). And after the tuning we achieved an RMSE of : 0.401.



We plotted predicted vs. actual log_bike_count values to assess the XGBoost model. Most points align with the red dashed line, showing accurate predictions, indicating the model effectively captures underlying patterns in the data, though slight dispersion at the extremes suggests reduced accuracy for very low or high values.
This project allowed us to deepen our understanding of handling missing values by addressing questions such as: How many values are missing? Are there duplicate rows? Is there a pattern in the missing data, and why are they missing? We explored various techniques to handle these issues. We also learned the importance of creativity in sourcing relevant external data and how to properly integrate it using the merge_asof function. Additionally, we gained insights into cyclical encoding, its application, and its limitations. The Exploratory Data Analysis (EDA) proved crucial in selecting the right predictors and identifying errors in the data, highlighting its role in improving model performance.

5 Possible alternative paths

A key limitation of our approach was using the counter ID as a feature in the model. A potential improvement would be to train separate models for each counter. This would transform the problem into a purely time-series task rather than a spatio-temporal one, better capturing counter-specific patterns since predictors can have varying effects across counters. Another promising direction would be to explore advanced time-series models such as LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units), which are types of recurrent neural networks (RNNs) particularly well-suited for time-series prediction. These models could effectively capture temporal dependencies and potentially improve prediction accuracy.

References

- Pandas (merge asof) : https://pandas.pydata.org/pandas-docs/stable/reference/ api/ pandas.merge_asof.html
- XGBoost Documentation : https : / / xgboost . readthedocs . io / en / stable/
- Sklearn. RandomForestRegressor : https://scikit-learn.org/stable/modules/generated/ sklearn.ensemble.RandomForestRegressor.html
- Scikit-learn. GridSearchCV. https://scikit-learn.org/stable/modules/generated/ sklearn.model_selection. GridSearchCV.html.
- Scikit-learn. OneHotEncoder Documentation: https : / / scikit - learn . org / stable / modules / generated / sklearn preprocessing.OneHotEncoder.html.
- Scikit-learn. Ridge Regression Documentation: https://scikit-learn.org/stable/modules/ generated/sklearn. linear_model.Ridge.html.