Project Report: Predicting Cyclist Traffic in Paris

Master of Science and Technology: Data Science for Business X-HEC

Solal Danan & Samuel Mesguiche

1 Data Preprocessing

1.1 Original Dataset

The initial phase in developing models to predict bike counts involved analyzing the provided dataset. The original dataset, with 455,163 observations and 9 predictors, contains information for various hours throughout the day from September 2020 to October 2021 across 56 counters. Although the dataset has no missing values, it includes redundant predictors such as counter name and counter ID, as well as intuitively irrelevant ones like counter installation date.

In terms of encoding strategies, we adopted specific approaches for different types of features. Cyclical encoding was found to be the most performant to encode months, days, hours. These time units exhibit a cyclical pattern. Figure 1 indeed illustrates the cyclical pattern that the log_bike_count for some counters follows throughout different hours. Cyclical encoding captures this periodicity, aiding the model in understanding time-related nuances in bike usage. Nonetheless, decision tree-based algorithms, constructing split rules one feature at a time, may fail processing sin/cos values together in the same split when using cyclical encoding. For counter name, we applied one-hot encoding [6], acknowledging the distinctiveness of each without any hierarchical order. Although GapEncoder [2] was experimented, one-hot encoding proved more efficient. Standardizing all numerical features with StandardScaler [9] enhanced the model's performance. Acknowledging the significant impact of counters on predicting log bike counts, we utilized a Target Encoder [10]. This technique assigns weights to each counter name based on its correlation with the log bike count, reflecting the observed disparity in bike counts across different counters. Finally, the creation of an interaction feature between month and hour was critical for understanding seasonal patterns in bike usage, as biking habits can vary greatly between different times and months.

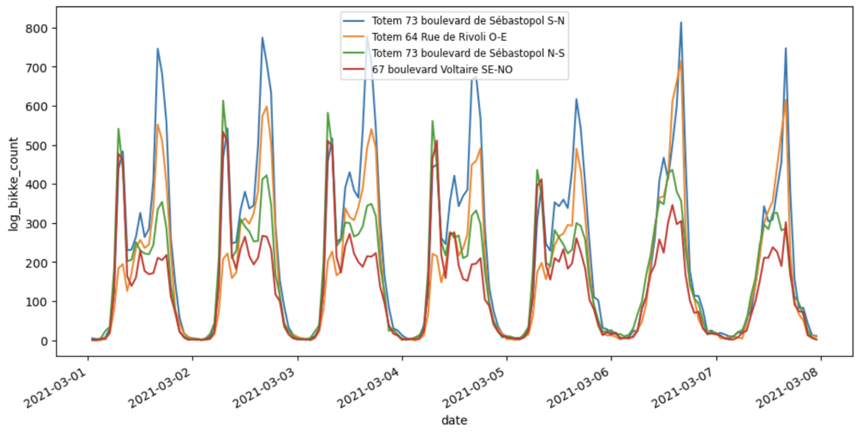

Figure 1: Variation in bike counts across hours over a week for four counters.

1.2 External Dataset

To complete our original dataset, we considered the given weather dataset obtained from the Paris-Orly weather station. This supplementary dataset contains 59 numerical features capturing weather conditions in Paris at 3-hour intervals from September 2020 to October 2021—the same time frame as our original dataset. Due to numerous missing values in this new weather dataset, we adopted a two-step approach for handling them. First, we eliminated features with more than 5% missing values, resulting in the exclusion of 31 features. Subsequently, we addressed the remaining missing values in other features by imputing them with the median value for each respective feature. We opted for the median over the mean to mitigate sensitivity to outliers.

In addition to the weather predictors, we manually introduced supplementary features to the external dataset, aiming to enhance the model and emphasize temporal aspects. We believed that boolean features indicating whether a given date falls within school holidays, bank holidays, the first or second quarantine, could significantly influence bike usage and contribute to the improvement of our model. Additionally, considering the noticeable decrease in counts during the Christmas holidays, as shown in Figure 2, we introduced a specific boolean column for this holiday period.

To merge the external dataset with the original one with the corresponding date, we employed the 'merge_asof' [4] function from the pandas library, essentially performing a left-join based on the nearest key rather than equal keys. Notably, both DataFrames had to be sorted by the key—in this case, the date—for this merging process.

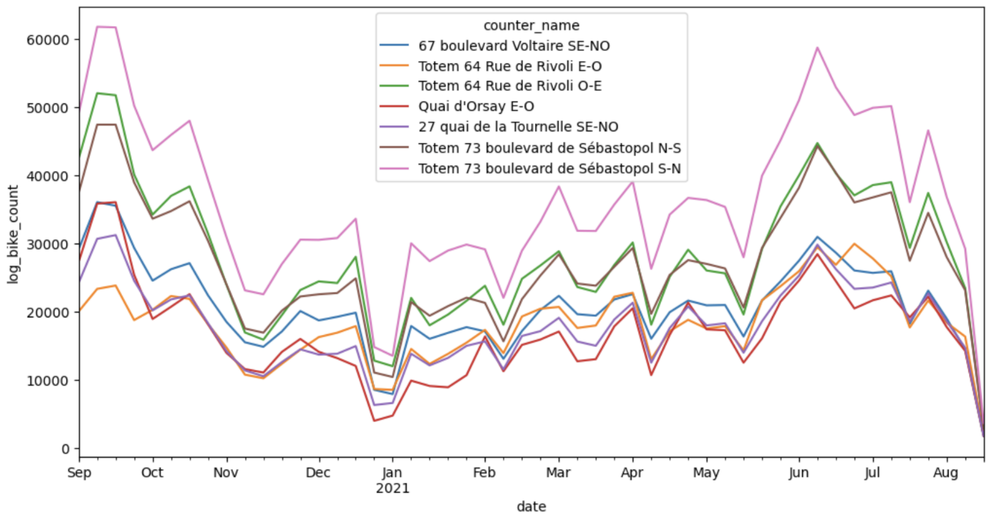

Figure 2: Plot of bike-counts per week against the date highlighting the downward trend over the Christmas holidays

2 Exploratory Data Analysis (EDA)

To assess the utility of various features and their relationship with the target variable (log_bike_count), we conducted exploratory data analysis using plots. Due to the volume of data, the initial plots proved illegible. To address this, we opted to visualize the mean of log-counts for each value of the predictor. The figure below provided valuable insights into which features merit inclusion in our model.

Several variables from the external dataset exhibited promising relationships, although not necessarily linear, with log_bike_count. Notably, "u" (humidity), "etat_sol" (soil condition), "cod_tend" (type of barometric trend), "ff" (average wind speed), and "t" (temperature). These initial plots provide a baseline for determining which predictors should be incorporated. However, we needed further testing for the remaining predictors because some of them look promising, such as "raf10" (gusts over the last 10min), "rafper" (gusts over a period), and "td" (dew point affecting rain).

Moreover, we analyzed the correlation between the features in our dataset, using the profile report, to see which features we should avoid using together (for instance, "rafper" and "raf10" or "w1" and "w2").

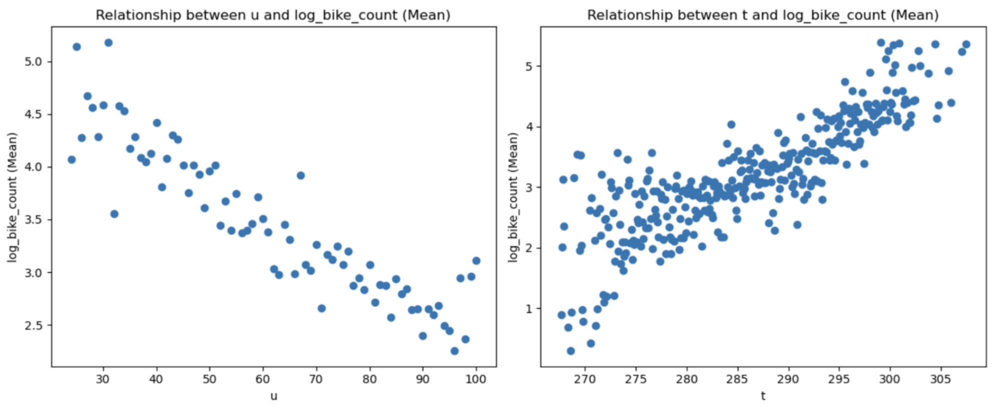

Figure 3: Plot showing linear relationship between features from the external dataset and the target variable

However, it's crucial to note that many predictors did not exhibit a linear relationship with the output variable. Therefore, if we were to employ Ridge regression [8] with all these predictors, it would be an inadequate model choice.

3 Model Selection

Ridge regression was the initial attempt in our modeling process. Nonetheless, decision tree-based algorithms, particularly Random Forest [7] and XGBoost [11], quickly outshone it. This result was expected, given the intrinsic abilities of these models to capture non-linearity and manage collinearity more proficiently than linear regression. Additionally, they benefit from mechanisms that mitigate overfitting.

Among the tree-based models, boosting algorithms showed superior performance. Their iterative approach to correcting errors, here the RMSE, contributed to their effectiveness and more accurate predictions. After XGBoost, LightGBM [3] was tested. However, it couldn't surpass the performance of CatBoost [1]. This aligns with benchmarks where CatBoost often outperforms other gradient boosting frameworks. Ultimately, CatBoost emerged as the best performing model. In refining our model, we experimented with including and excluding various external features. The best results were achieved by retaining the selected features bellow. To optimize the model, GridSearchCV [5] to tune hyperparameters was used and this process was particularly focused on the CatBoost model, given its superior performance.

After experimenting with various feature columns across different models and adjusting the hyperparameters, our most promising model emerged as a CatBoostRegressor with the following hyperparameters: learning_rate = 0.095, depth = 10, iterations = 5000, l2_leaf_reg = 5, bagging_temperature = 2.

The features utilized in this final model were: . *'month', 'day', 'hour', 'year','weekday', 'counter_name', 't', 'u', 'ff', 'tend', 'raf10', 'etat_sol', 'vv', 'latitude', 'longitude', sin_month, cos_month, sin_day, cos_day, sin_hour, cos_hour, counter_name_weighted, interation_hour_month, quarentine_1, quarentine_2, holiday, christmas_holiday, bank_holiday.*

4 Model Evaluation

From figure 4, we can infer that the model demonstrates a strong positive correlation between the predicted and actual target, indicating a good fit. The concentration of points along the diagonal suggests the model's predictions are mostly accurate. However, the presence of vertical stripes may indicate that for certain true values, the model predicts a wide range of counts, signifying potential variance in predictions. This is particularly the case when the log bike count is low which means that our model has more accurate predictions for counters with high counts. This could be due to outliers, or certain influential factors not being captured adequately by the model.

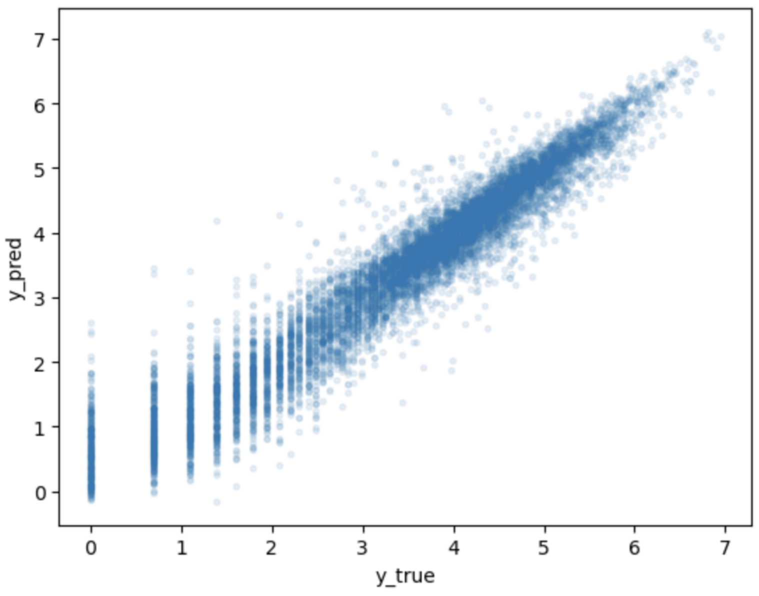

Figure 4: Scatter Plot of the RMSE of the predicted target variable against its actual value

Figure 5 provides insight into the model's performance over time, comparing the predicted bike counts against the actual counts across different days. If the orange dashed line mirrors the blue line closely, it confirms the model's strong temporal predictive capability. Nonetheless, any consistent gaps between the two lines would point towards systematic prediction errors at specific times. We note that the model gives poor predictions for some periods on the month of September. It might be improved by further investigation, feature engineering or model adjustments.

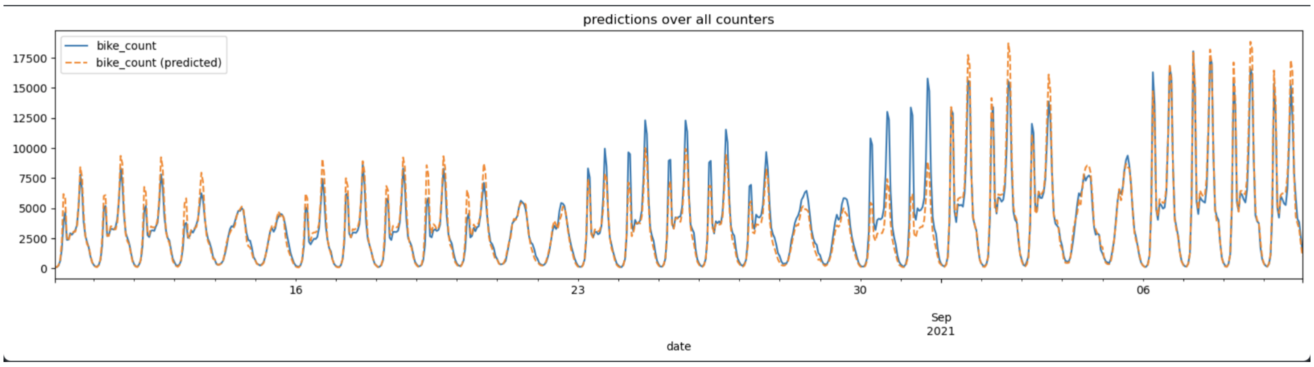

Figure 5: Plot showing the model's prediction performance

5 Potential refinements

Firstly, the model demonstrates a tendency to perform more accurately when predicting higher log bike counts as shown in Figure 4. It may suggest a sensitivity to outliers, where certain counters exhibit abnormally low bike counts during specific periods. Such anomalies could arise from various factors, such as temporary road closures, or construction work. The presence of these outliers implies that our model may be more reliable during stable, high-activity periods. To address this, a more thorough investigation into the data would be needed. Identifying and understanding these outliers could lead to refined preprocessing steps, such as outlier detection and mitigation. By doing so, we could enhance the model's robustness, ensuring it remains accurate and reliable, even when anomalies occur.

Further, incorporating additional datasets could substantially improve prediction. For instance, integrating data on Paris's street population density or vehicular traffic near the counters could capture the dynamic interplay between cyclists' behavior and their environment. A lack of time and resource availability have limited our capacity to explore these options fully.

Finally, exploring neural networks could have yield interesting results. Neural networks are adept at capturing nonlinear relationships and interactions in high-dimensional data, which could be particularly useful given the varied and rich dataset at our disposal.

References

[1] CatBoost developers. *CatBoost*. https://catboost.ai.

[2] DirtyCat developers. *GapEncoder Documentation*. https://dirty-cat.github.io/stable/generated/dirty_cat.GapEncoder.html.

[3] LightGBM developers. *LightGBM Documentation*. https://lightgbm.readthedocs.io/en/latest/Python-Intro.html.

[4] Pandas. *pandas.merge_asof*. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.merge_asof.html.

[5] Scikit-learn. *GridSearchCV*. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[6] Scikit-learn. *OneHotEncoder Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.

[7] Scikit-learn. *RandomForestRegressor*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[8] Scikit-learn. *Ridge Regression Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html.

[9] Scikit-learn. *StandardScaler Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[10] Scikit-learn. *TargetEncoder Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.TargetEncoder.html.

[11] XGBoost developers. *XGBoost Documentation*. https://xgboost.readthedocs.io/en/stable/.