BIKE COUNTERS PROJECT: FINAL REPORT

MAP536: PYTHON FOR DATA SCIENCE

Sophie Kitchin and Manan Gupta

December 2021

Contents

1 Introduction

In this paper, we outline the methodology we used to build a model to predict the number of bikes passing bike counting stations in Paris each hour. We begin with a description of the data, including additional features that we found in external sources. Next, we summarise the exploratory data analysis that we conducted, before describing the feature engineering and missing value imputation methods we made use of. Then we summarise and compare the models that we used, and analyse our results, giving some interpretation. Finally, we conclude that our best model was a tuned version of a regularised gradient descent algorithm (XGBoost), and summarise how we optimised our model.

2 Data

We were provided with 496 771 observations from 30 different cyclist counting sites installed around Paris by the city council. The data ranged from the 1st September 2020 to the 9th September 2021. Our target variable was the log bike count and the starting kit for the project contained the following features: counter name, counter site name, date, counter installation date, latitude, and longitude.

3 External Data

Data	Description of features extracted	Type and source*	Reason for using
National holidays	Binary indicator of whether each day was a national holiday in France.	Categorical, French government website	We hypothesised that the number of people travelling by bike may show a different trend on national holidays compared to normal working days and weekends.
Weather data	40 weather-related variables, see Appendix 2 for a full variable list.	Quantitative, Meteo France	Despite Orly being 15km from Paris and the source only providing 3-hourly data, we were unable to find more localised, granular data for free, so we used the external data source provided. Orly is geographically close enough to our bike counter locations to still provide predictive power.
Covid Data	Daily covid cases and vaccinations in the Paris region.	Quantitative, French government website	We added this data as the rapidly increasing Covid cases in the second and third wave were likely to discourage people from going out, leading to reduced bike traffic over these periods. Equally, we hypothesised that vaccinations may encourage people to go out again, due to increased immunity.
Mobility Data	Tracks percentage change in average time spent in certain locations with a baseline established before the pandemic.	Quantitative, Google	We hypothesised that many people would have changed their mobility habits in response to the pandemic. Therefore we included data on changes in data time spent in: residential, grocery, workplace, stations, parks and retail locations.
Pollution Data	Levels of pm2.5 and pm10 particulate matter.	Quantitative, AQI index for Paris	Pollution may discourage people from cycling or else be informative as a proxy for how many polluting vehicles are using the road or the weather conditions.
Traffic Data	3 measures of traffic for motorised vehicles (occupancy rate, flow rate and vehicle number).	Quantitative, Paris Open Data	We hypothesised that using other traffic data my help us better predict the extreme spikes observed in our target variable. We found data on the presence of vehicles on fixed loops, number of vehicles and a flow of vehicles per unit time.
Lockdown Severity	Continuous measure of lockdown severity in France.	Quantitative, Our World in Data	Initially, we found a binary indicator, but later found this richer continuous measure.

$*$ Please see Appendix 2 for a web link.

We added the features in the above table to our dataset in order to augment the data provided in the starting kit in, and gain predictive power.

4 Exploratory Data Analysis

Once we had collected and cleaned our data, we conducted an exploratory data analysis. This included visualising the spatial and temporal distribution of our data. Firstly, to better understand the bike counters information we plotted the average of the bike counters variable against time (see Figure 2 of Appendix 1). From this, we can see a clear seasonal trend (with fewer bikes counted in winter), a weekly trend (with less cyclists at the weekend than mid-week), and daily trend (with two spikes in the morning and evening).

Our temporal analysis confirmed our initial hypothesis that the bike count would be negatively correlated with the Covid stringency index and the reduction in time spent at the workplace (from Google mobility data), as when the lockdown was tighter we reasoned that less people would be commuting to work, or elsewhere, via bike (see Figure 3 of Appendix 1 for a coherent picture of the impact of lockdown using three different data sources).

Additionally, we explored the correlations between all variables in our dataset (see Figure 4 and 5 of Appendix 1), to understand which were most correlated with our target (log bike count). We found that vehicle number, temperature (t) and humidity (u) were most highly correlated with the target, so might be important features to include in our model.

5 Feature engineering and pre-processing

5.1 Trigonometric encoding of time

We used our exploratory data analysis to help us conduct feature engineering. Our exploratory data analysis revealed that our target variable was highly dependent on daily/weekly cycles and yearly seasonal cycles. Raw encoding of the time variable could prevent models from recognising the heterogeneous impact of one additional hour at different times of day. For this reason, we decided not to encode time in a simple monotonic manner using ordinal encoding, but instead experimented with one-hot encoding to add flexibility. The disadvantage of this approach was that it ignores the ordering of time variables which could prove informative. Therefore, we encoded each periodic time feature using a sine and cosine transformation. This encoding preserves the cyclic nature of the time variable, especially visible in the hour variable (see Figure 6 and 7 of Appendix 1). Moreover, an advantage of this approach is that there is no jump between the first value in the periodic range and the last, which makes logical sense as 1am follows midnight sequentially.

5.2 Missing data imputation using KNN

The external data gathered about the vehicle flow and traffic occupation had a large amount of missing data as there were some hours during the day where the counters were not functioning or did not gather enough data for the readings to be available. This is not ideal as missing data can reduce the ability of the algorithm to accurately predict the trends and patterns.

To fill in this missing data, we could use a simple method such as filling in the missing values with the mean of the columns. However, this is not suitable in this case as filling in the values with the means will not capture the traffic patterns accurately. Therefore, we decided to use KNN Imputation to fill in these missing values. This imputer utilises the k-Nearest Neighbor's method to replace the missing values in the datasets with the mean value from the parameter "n_neighbors" nearest neighbours found in the training set. By default, it uses a Euclidean distance metric to impute the missing values. The method KNN imputation therefore allows us to capture the variability and patterns of the missing data and is a lot more precise than just using the means of the columns.

6 Models

6.1 Linear Model

A widely used model is linear regression, but when we implemented a regularised version (ridge regression) and tuned the hyperparameters our RMSE remained quite high, at around 0.80, so we quickly opted to move onto models that would fit our non-linear data better.

6.2 Gradient Boosting Regressor

Gradient boosting is a decision tree-based ensemble machine learning algorithm, which tends to work well on tabular data. However, gradient boosting can be slow to train as trees are added sequentially.

6.3 Histogram Gradient Booster

An extension of gradient boosting, which places continuous features into discrete bins and uses these bins to construct histograms during training. Histogram gradient boosting tends to produce faster algorithms that use memory more efficiently than gradient boosting.

6.4 Extreme Gradient Booster

XG Boost (Extreme Gradient Boosting) is again a decision tree-based ensemble algorithm that uses an optimised implementation of gradient boosting. It has built in L1 (lasso) and L2 (ridge) regularisation to prevent over-fitting. Furthermore, XGBoost makes use of parallel processing so it is much faster than other gradient boosting algorithms.

Another advantage of XGBoost is that it makes splits until the specified max depth and then prune trees backwards to remove splits beyond which there is no positive gain. This is in contrast to traditional gradient boosting, which is a more greedy algorithm, and would simply stop splitting a node as soon as it encounters a negative loss in that split.

6.5 Hyperparameter Tuning

Hyperparameter tuning involves dealing with many inherent tradeoffs within the model, notably to find the optimal bias-variance tradeoff, thus controlling for overfitting of the model and ensuring robustness. For this reason, we tuned the number of trees, which informs the algorithm when to stop to avoid overfitting and the max tree depth, which contributes to controlling the tradeoff between overfitting and robustness. Increasing our lambda and alpha regularisation parameters is yet another method we used to reduce variance and avoid an overly complicated, unstable model. Minimum child weight is another regularisation parameter, which tells the algorithm to stop trying to split once the sample size at a node goes below a given threshold. Meanwhile subsample indicates the proportion of observations to be randomly sampled for each tree, with smaller values being used to prevent overfitting.

Meanwhile, the learning rate results in a higher precision when it is lower, but also an increased training time for the model. In this way, we tuned many hyperparameters simultaneously to find the optimal combination of hyperparameters to balance these tradeoffs, as can be seen in the "estimator.py" file of our final model.

Initially, we used randomised search cross validation to get close to the optimal parameter values and then used grid search cross validation to further optimise the many hyperparameters in our models. However, the training time was a practical issue with this approach. Furthermore, randomised and grid search treat experiments on hyperparameters as independent. Therefore, we later used Bayesian hyperparameter optimisation to tune our models' hyperparameters. Bayesian methods use sequential model-based optimisation, so the model learns from previous iterations which hyperparameters to try next. By looking in the relevant search space, Bayesian optimisation can thus minimise the tuning time, which allowed us to tune a large number of parameters at once.

7 Results

We scored our models using RMSE, and our XGBoost model produced the best results using cross validation on the validation dataset, with an RMSE of 0.667 and, therefore comprised our final model.

Our model fitted the actual number of bikes reasonably well, but still had trouble predicting the peaks in the number of bikes. However, plots showed that XGBoost dealt better with predicting the peaks in bike traffic compared to histogram gradient boosting (see Figure 1 for our predictions from these two different models on the test set).

Table 1: RMSE on validation set with different tuned models

Model	RMSE
Ridge Regression	0.896
Gradient Booster	0.746
Histogram Gradient Booster	0.727
Extreme Gradient Booster	0.667

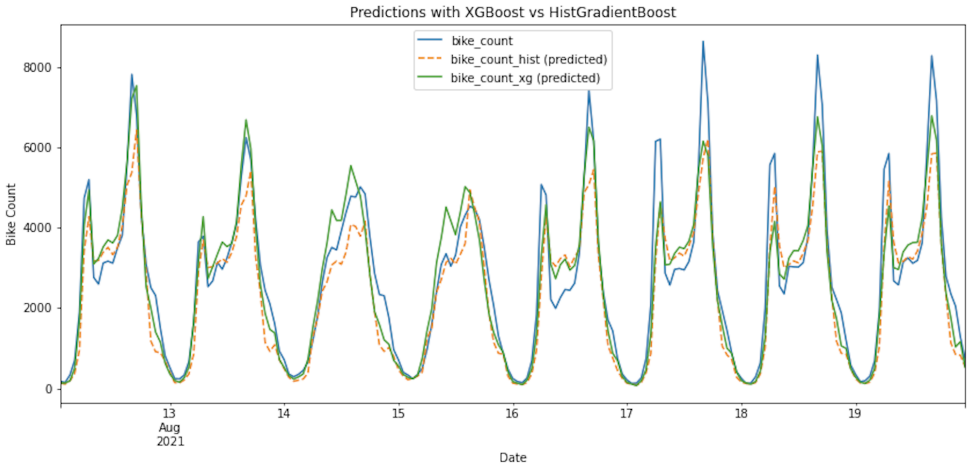

Figure 1: Model predictions with histogram and extreme gradient boosting and actual bike count.

An advantage of using XGBoost, is the interpretability of the results. The feature importance shown in Figure 8 of Appendix 1 reflects the average gain across all splits when that feature was used to split the data. From this we can see that the temperature, hour of the day, Google mobility data and weekday were all important variables that added significantly to the model when data was split on them.

We also computed the permutation importance for our final model (Figure 9 of Appendix 1). This calculates the increase in the model's prediction error after permuting each feature. A feature is important if shuffling its values increase prediction error a lot. Results were broadly aligned with the feature importance by average gain, and confirmed that the hour, number of vehicles, mobility data and weekday were all important features in our model.

8 Conclusion

Our results demonstrate that XGBoost's regularised gradient boosting model can perform well on medium-sized structured datasets. Moreover, trigonometric encoding of time, KNN imputation of missing data and Bayesian optimisation methods significantly improved our model. Furthermore, our analysis of feature importance in our final model reaffirms the importance of many of the additional features that we found in external data sources.

9 Appendix 1: Figures

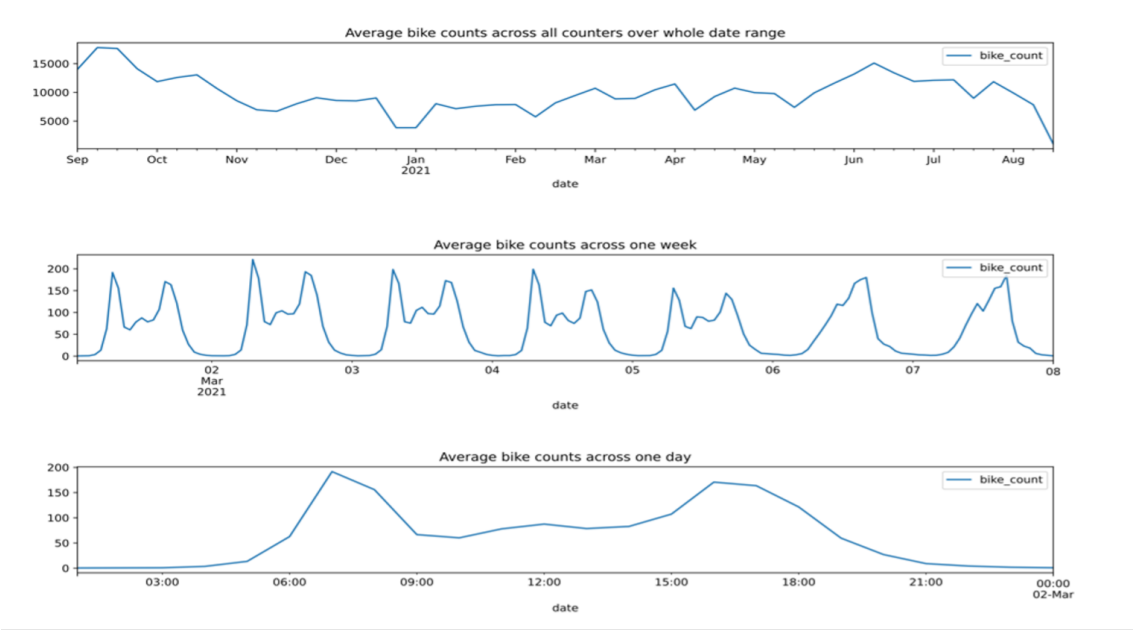

Figure 2: Time trend of bike counter variable.

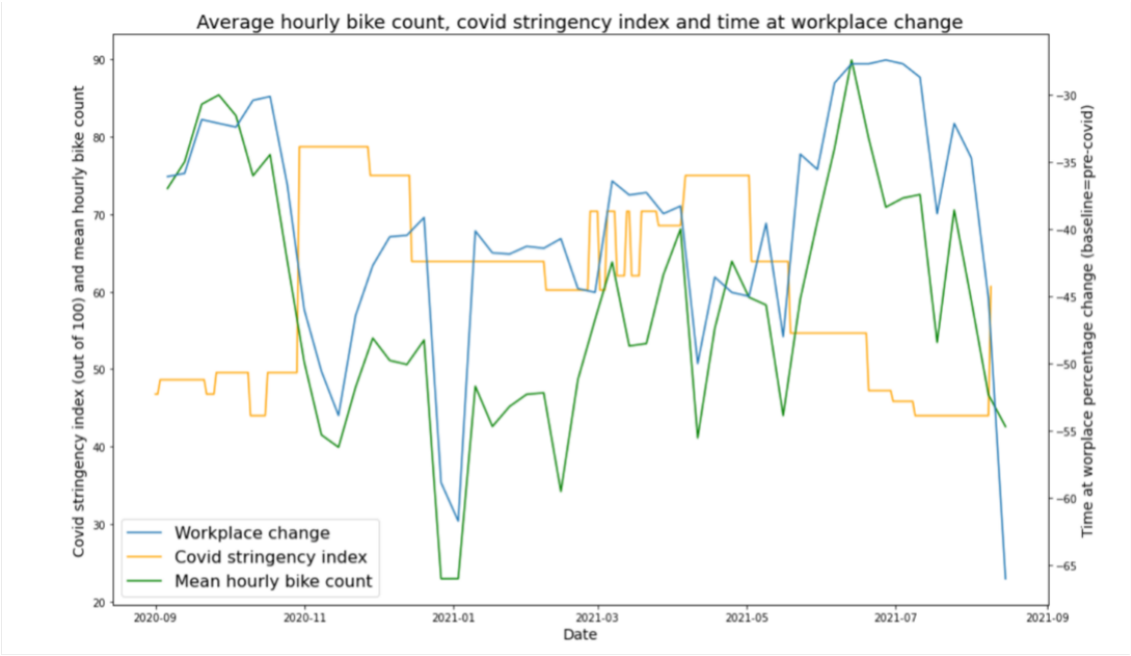

Figure 3: Change in average bike count, Covid stringency and time spent in the workplace over time.

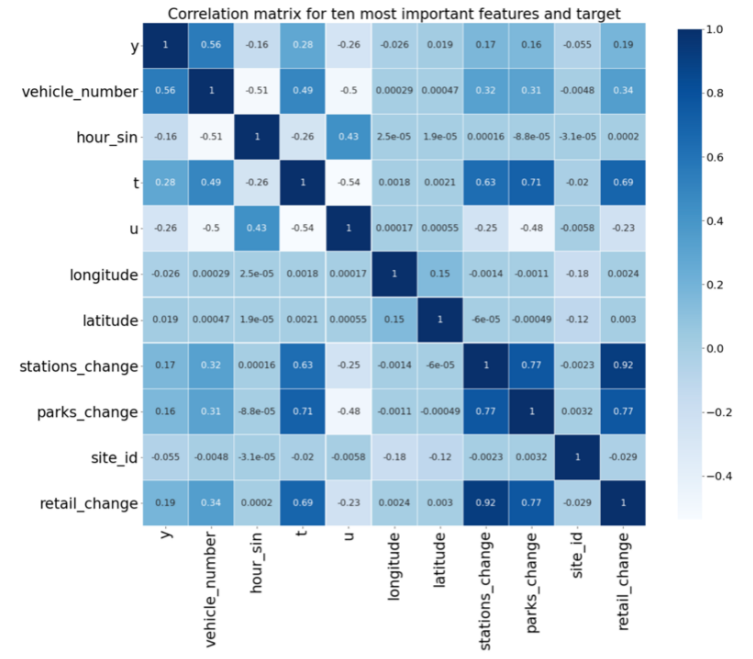

Figure 4: Correlation matrix for 10 key features and target variable.

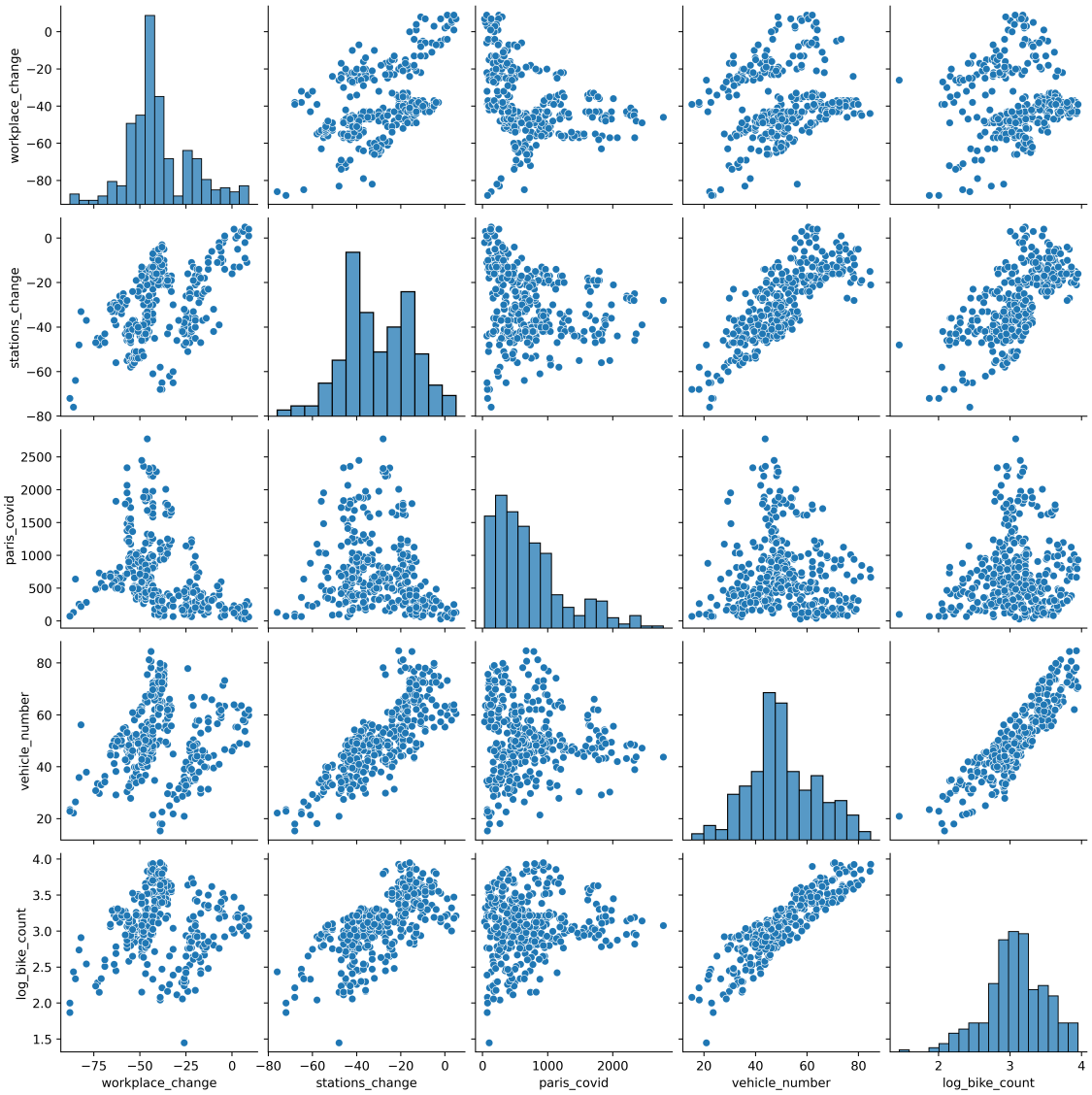

Figure 5: Correlation plots for key variables from exploratory data analysis.

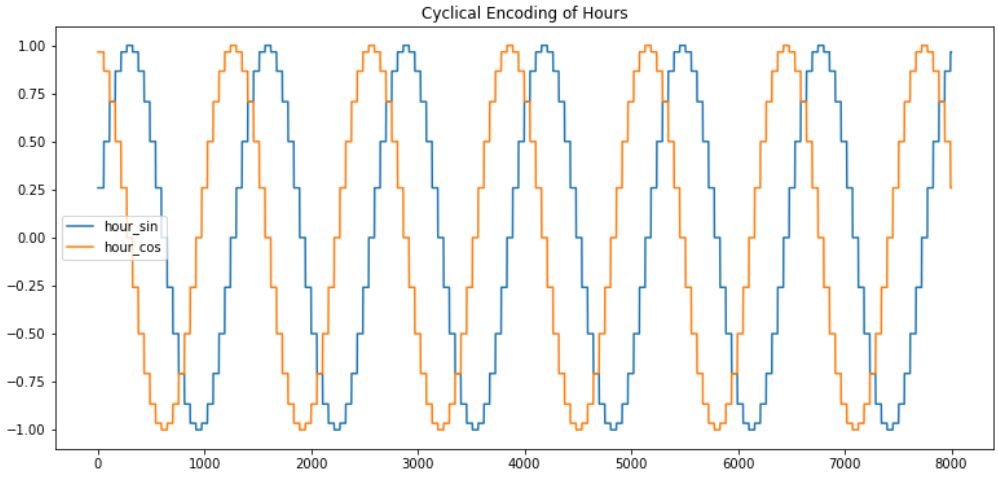

Figure 6: Sine and Cosine encoding of hour variable.

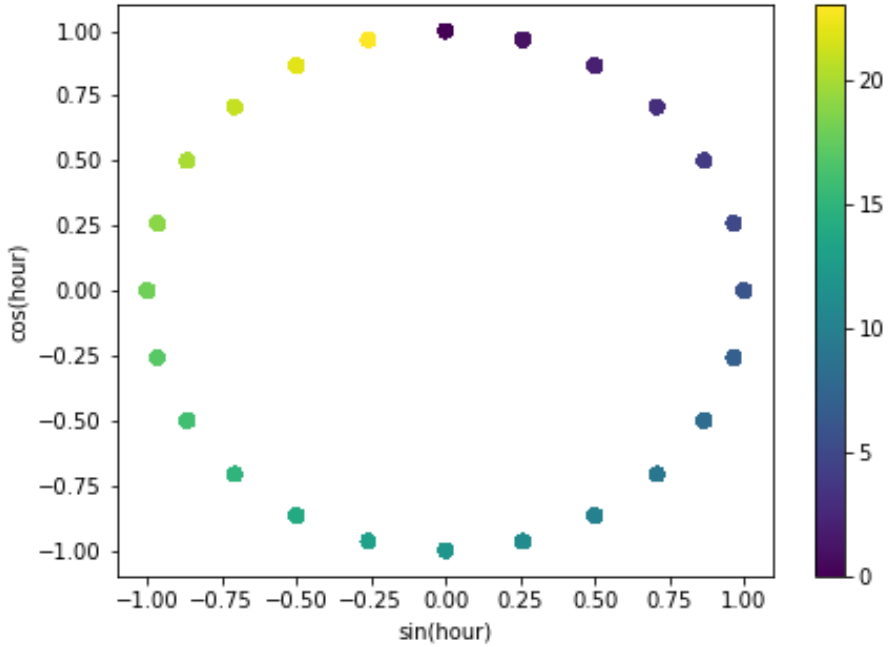

Figure 7: Encoding of hour variable.

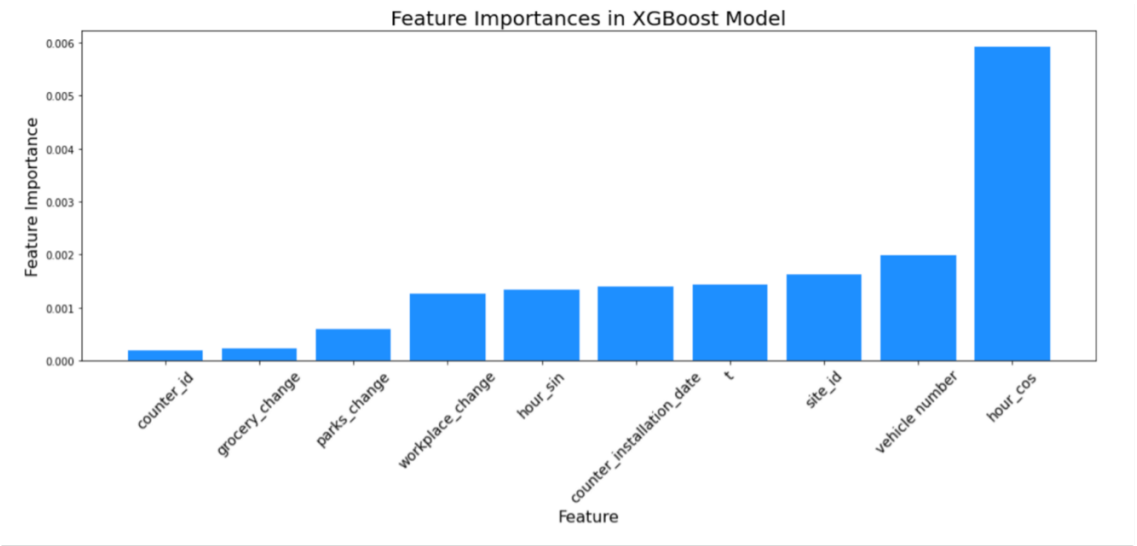

Figure 8: Feature importance from extreme gradient booster model.

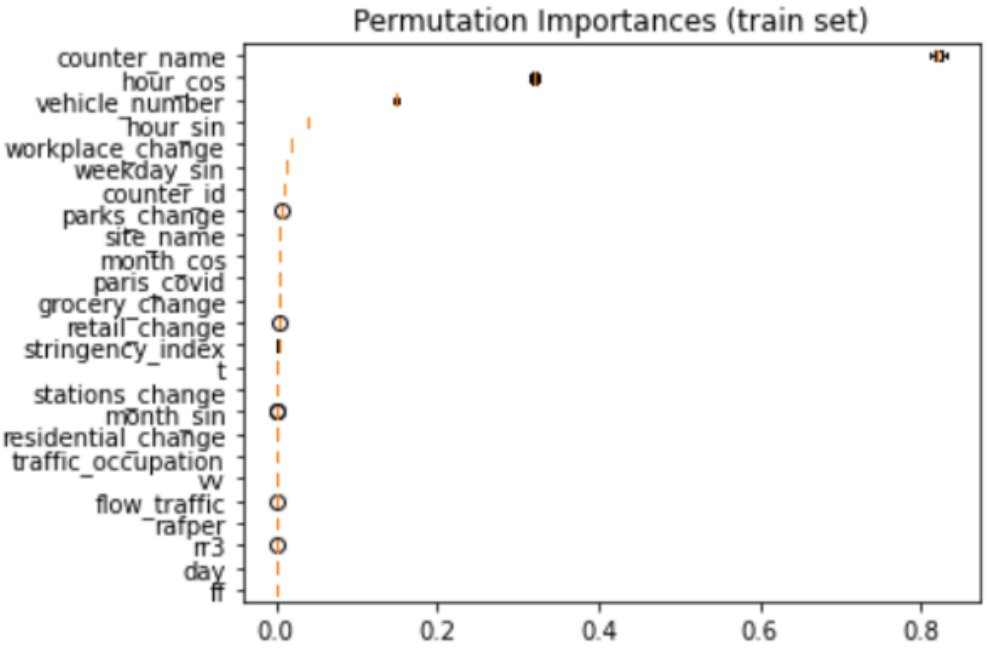

Figure 9: Permutation importance for extreme gradient booster model.

10 Appendix 2: Sources

1. Weather Data:https://meteofrance.com
2. Weather variable list: `https://donneespubliques.meteofrance.fr/client/document/doc_parametres_synop_168.pdf`
3. National Holidays Data :https://www.data.gouv.fr/en/datasets/jours-feries-en-france/
4. Covid Stringency Data:https://ourworldindata.org/grapher/covid-stringency-index
5. Traffic Data:https://opendata.paris.fr/explore/
6. Mobility Data:https://www.google.com/covid19/mobility/
7. Covid Data:https://www.data.gouv.fr/fr/
8. Pollution Data:https://aqicn.org/city/paris/
9. Lockdown Severity Data :https://ourworldindata.org/grapher/covid-stringency-index
10. Manan and Sophie RAMP Project Github repository:`https://github.com/manang1803/Ramp_Project_Manan_Sophie/`