Machine Learning Fall 2023
Final Project: Regular Track

Team: Alan 不講-.-

Learners: R12522820 洪柏濤 R12631055 林東甫 R12522808 黃正渝

Data Set

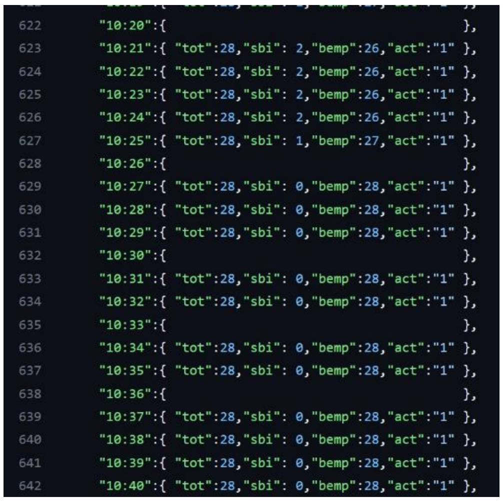
Dataset update daily to show the latest data collection status, and there're some issues with raw data:

I. **Missing values:** The raw data contains a significant number of missing values, frequently occurring at time points ending in 00, 10, 20, 30, 40, and 50.

II. **New station update:** There are occasional additions of stations at irregular time points, and the content data for these stations is often incomplete.

III. **Data ingestion times**: The process of importing raw data is time-consuming.

Hence, data preprocessing is critical to address these problems.



```
"10:20":{                                                    },
"10:21":{ "tot":28,"sbi": 2,"bemp":26,"act":"1" },
"10:22":{ "tot":28,"sbi": 2,"bemp":26,"act":"1" },
"10:23":{ "tot":28,"sbi": 2,"bemp":26,"act":"1" },
"10:24":{ "tot":28,"sbi": 2,"bemp":26,"act":"1" },
"10:25":{ "tot":28,"sbi": 1,"bemp":27,"act":"1" },
"10:26":{                                                    },
"10:27":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:28":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:29":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:30":{                                                    },
"10:31":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:32":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:33":{                                                    },
"10:34":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:35":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:36":{                                                    },
"10:37":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:38":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:39":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
"10:40":{ "tot":28,"sbi": 0,"bemp":28,"act":"1" },
```

▲ Lots of missing values in raw data.

Data Preprocessing

1. **Create a CSV file for each day of the prediction period:** To facilitate the writing of daily data, we first create CSV files for each day from the prediction start date (2023/10/02) to the prediction end date (2023/12/17). The file name should be the date, such as ex: 20231002.csv. The first row of each file should consist of *'id'*, *'tot'*, *'sbi'*, *'bemp'*, *'act'* to facilitate the later writing of daily data.

2. **Obtain station:** Load *'demographic.json'* as *'demo'*. The keys of *'demo'* are names of each station *('sno')*.

3. **Define the function *writeDailyData(date)*:** For a given input variable *date*, check if the file exists. If it does, read the JSON file and fill missing values with the previous value using *fillna(method='ffill')*, concerning the time-dependent nature of bicycle quantity changes. Considering the possibility of the first value being missing, use *fillna(method='bfill')* to fill missing values with the next value. Use *filter(regex='[024]0$')* to filter out the required data for 20, 40, and 60 minutes of prediction. Finally, use writerow() function to write the filtered data to the respective CSV, where *'id'* corresponds to the column data format date_sno_time, such as *20231002_500101001_00:00*, and *'tot'*, *'sbi'*, *'bemp'*, *'act'* represent the corresponding data for that id.

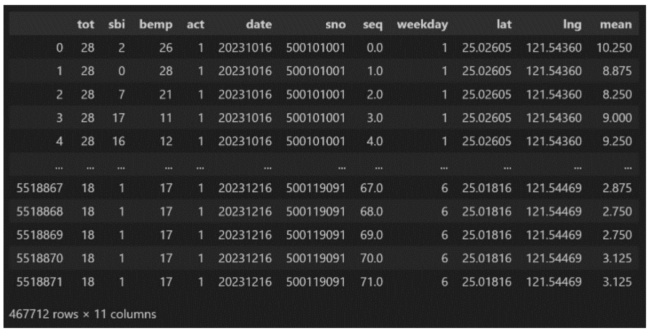
4. **Define the function *writeTimes(startDate, endDate)*:** For given input variables *startDate* and *endDate*, repeat the above *writeDailyData(date)* function from the start date to the end date to automate the process of writing daily data and become much more efficient.

5. **Write files:** Specify start date *(startDate)* and end date *(endDate)*, then execute *writeTimes(startDate, endDate)* function to preprocess the data quickly, converting the daily JSON files into a single CSV file.

6. **Obtain a new feature - time series *'seq'* & *'weekday'*:** Extract information from the 'id' column to create new columns *'date'*, *'sno'*, and *'time'*. Split the *'time'* column to obtain *'hr'* and *'min'* information, converting the original time to a time series *'seq'*. For example, 00:00 becomes 0, 00:20 becomes 1, ..., and 23:40 becomes 71. Remove the unnecessary columns *'id'*, *'time'*, *'hr'*, *'min'* using *drop(axis=1)*. Same way to extract from the *'date'* column and add a new column *'weekday'* representing the day of the week.

7. **Add the feature - latitude and longitude *'lat'*, *'lng'*:** Read the latitude and longitude of each station from the original *'demographic.json'* and add two new columns.

After preprocessing, data becomes as in the right figure.



	tot	sbi	bemp	act	date	sno	seq	weekday	lat	lng	mean
0	28	2	26	1	20231016	500101001	0.0	1	25.02605	121.54360	10.250
1	28	0	28	1	20231016	500101001	1.0	1	25.02605	121.54360	8.875
2	28	7	21	1	20231016	500101001	2.0	1	25.02605	121.54360	8.250
3	28	17	11	1	20231016	500101001	3.0	1	25.02605	121.54360	9.000
4	28	16	12	1	20231016	500101001	4.0	1	25.02605	121.54360	9.250
...	...	...	...	...	...	...	...	...	...	...	...
5518867	18	1	17	1	20231216	500119091	67.0	6	25.01816	121.54469	2.875
5518868	18	1	17	1	20231216	500119091	68.0	6	25.01816	121.54469	2.750
5518869	18	1	17	1	20231216	500119091	69.0	6	25.01816	121.54469	2.750
5518870	18	1	17	1	20231216	500119091	70.0	6	25.01816	121.54469	3.125
5518871	18	1	17	1	20231216	500119091	71.0	6	25.01816	121.54469	3.125

467712 rows × 11 columns

Exploratory Data Analysis

When first time received this dataset, we had some conjectures about it:

I. **Time-Period:** Given the regular daily routines of users, the data may exhibit strong periodicity.

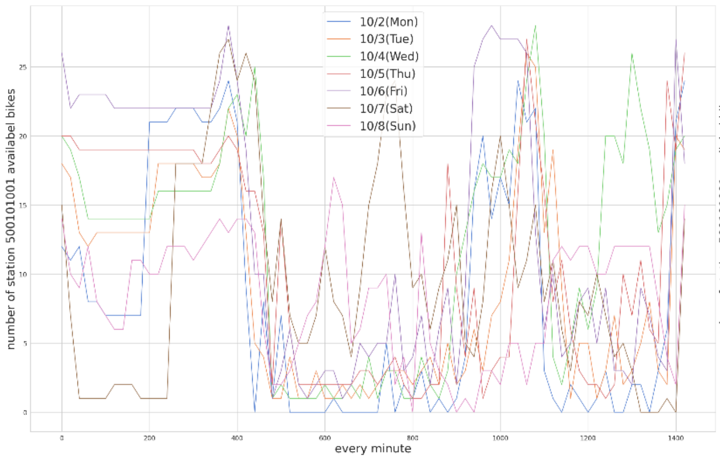
II. **Geographic Location:** Considering that users tend to search for available bikes in nearby locations, geographically adjacent stations may exhibit strong correlations.

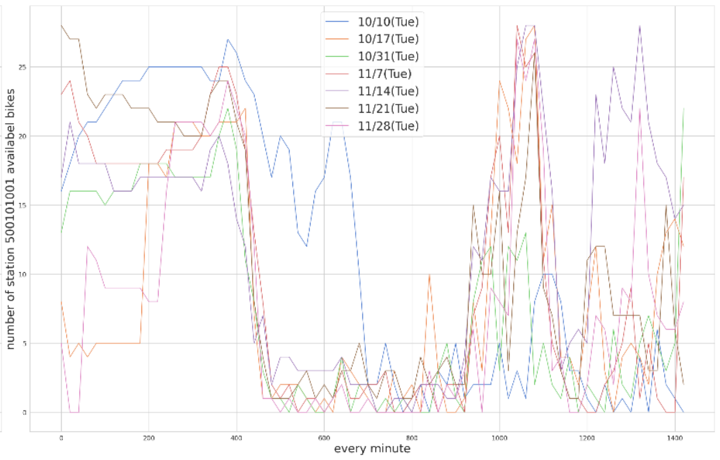
III. **Seasons, Weather & Holidays:** The usage of bikes as a way of transportation may also be influenced by the current season or weather conditions and whether it is holiday or not.

Based on the aforementioned conjectures, we have decided to conduct some preliminary analysis on the data.

First, we have selected the daily availability of a single station to observe, including daily and weekly:

The left side of the chart represents ‘sbi’ over time for different days within a week (10/2~10/8) at ‘500101001’. The right side represents ‘sbi’ over time on Tuesday (10/10, 17, 31, 11/7, 14, 21, 28) in different weeks.

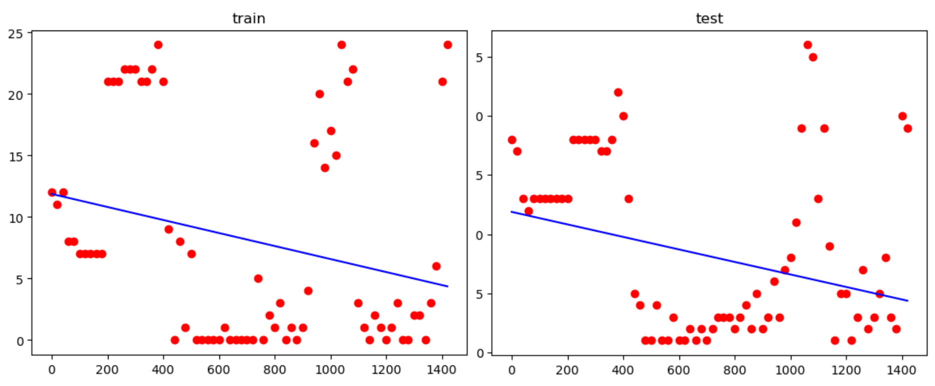




As shown in the graph, it could be evident that the data exhibits strong daily and weekly periodicities. This might be two types of cycles, with the bigger cycle possibly corresponding to a weekly pattern (from Mon. to Sun.), while the smaller cycle aligns with a daily pattern (from 0am to 23pm).

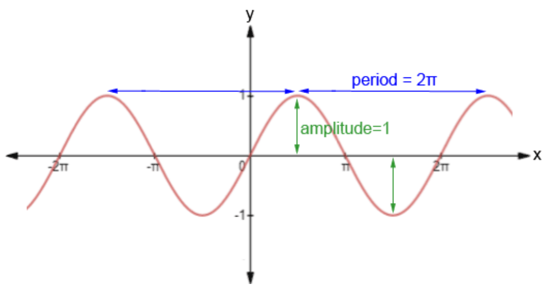
It is notable that October 10th stands out, likely due to being a holiday. Consequently, its data distribution differs from other Tuesdays. Such data may provide useful insights for predicting holiday usage in the future.

Finally, we utilized existing packages (scikit-learn) to split the available data into training set and testing set and conducting a simple linear regression analysis. However, the results were not promising, as shown above, with initial r-square scores even falling below 0.1. This suggests that simple linear regression may struggle to capture the patterns in this complex type of time series dataset.



Feature Engineering

In the EDA, we observed that the data may exhibit two types of cycles: a smaller daily cycle and a larger weekly cycle. To highlight the periodic cycle features of the data, we first extracted the time component and then combined it using the sine function. Formula of daily and weekly cycle as follows [1]:



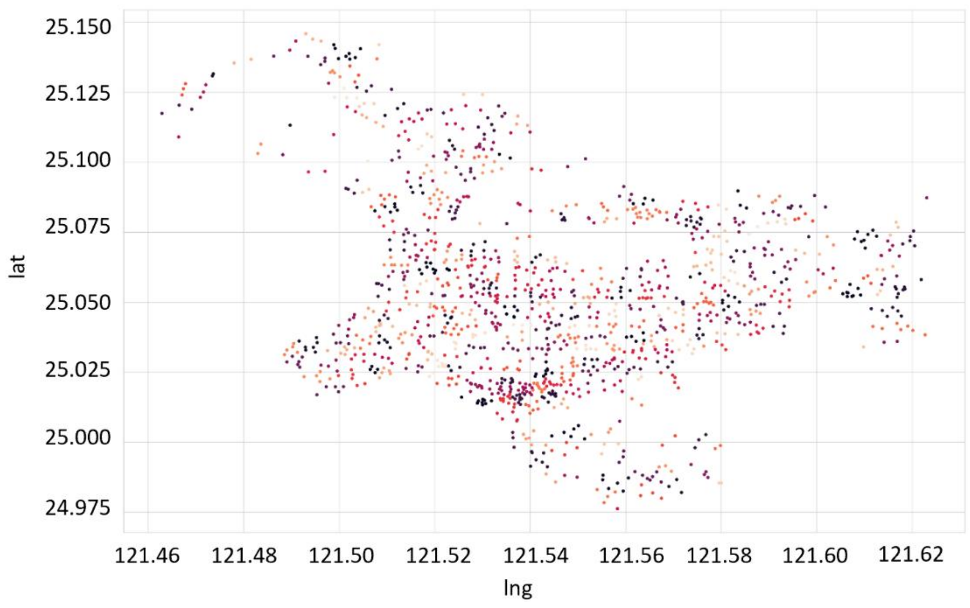
$$w(d) = \sin\left(\left(\frac{d}{7}\right) * 2\pi\right) \ where\ d = 0, 1, 2, 3, 4, 5, 6 \qquad d(m) \sin\left(\left(\frac{m}{72}\right) * 2\pi\right) \ where\ m = 0, 1, 2, \dots, 70, 71$$

As a result, we turn the day-of-week information into a cyclic continuous value:

	Mon.	*Tue.*	*Wed.*	*Thu.*	*Fri.*	*Sat.*	*Sun.*
Sin(x)	0.78183	0.97492	0.43388	-0.43388	-0.97492	-0.78183	0.0

The same approach was also applied to the daily cycle at 20-minute intervals.

Next, to obtain spatially adjacent features, we employed the k-means algorithm to categorize points with close geographical proximity into the same group, assigning them identical numeric labels. Closer numeric values indicate closer geographical locations. [2] (k value = 300 was determined through a brute-force approach.)



Finally, to combine both the weekly and spatial periodicities into new features, we calculated the average for each station at every time point on the same day of the week. In other words, we performed mean encoding [3] for each station, day of the week and time:

- Mean of Station-Weekday-time: Mean of '500101001'-Sun-00:00: 9.777777777777779
- Mean of Station-Weekday-time: Mean of '500101001'-Sun-00:20: 7.555555555555555

……

- Mean of Station-Weekday-time: Mean of '500119091'-Mon-23:20: 5.888888888888889
- Mean of Station-Weekday-time: Mean of '500119091'-Mon-23:40: 3.666666666666667

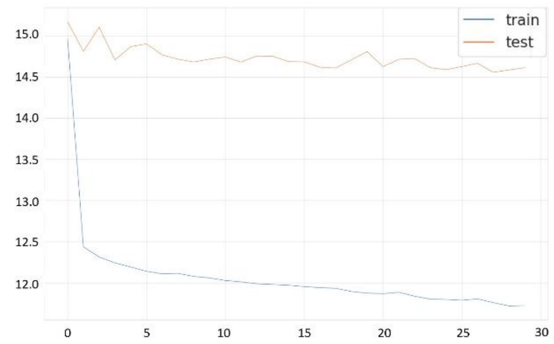
Model Selection

When dealing with time series data like this, aside from specialized statistical tools like ARIMA, one of the first considerations is often Long Short-Term Memory (LSTM) networks. LSTM is a type of recurrent neural network known for its ability to capture long-range dependencies in sequential data. Its key feature is the use of memory cells that selectively retain and update information, making it well-suited for modeling and predicting complex temporal patterns. To save development time, we directly employed the LSTM model from Keras. In the preliminary phase, only the data of 112 stations of prediction was considered. Despite this, training the model on just one month's data still took over 2 hours (with one 4090 environment), the obtained regression score (coefficient of determination, R^2) was approximately around 0.5.

Next, we attempted to use gradient boosting models. This kind of models are ensemble learning techniques that build a strong predictive model by combining the outputs of multiple weak models, often decision trees. They sequentially correct errors of preceding models, making them robust, interpretable, and effective for various tasks, especially in regression and classification problems. Despite the awareness that gradient boosting models may not be the ideal choice for time series prediction, but the score is still comparable to that of LSTM.

Finally, our third model is also an ensemble learning approach. In order to differentiate it from the second model, we use the random forest algorithm to construct the model. Unlike gradient boosting models, random forest belongs to the bagging logic. It creates new sampled training datasets by resampling with replacement from the original training dataset. This step is repeated n times to create n different decision trees, and these decision trees are used for prediction. For regression problems like predicting the number of bicycles, predictions can be averaged, or for classification problems, the majority generated by the voting method can be used to make decisions. Due to its simplicity, fast training speed, and strong generalization ability, we believe it can be effectively used as a prediction model for this project.

Fine-tuning



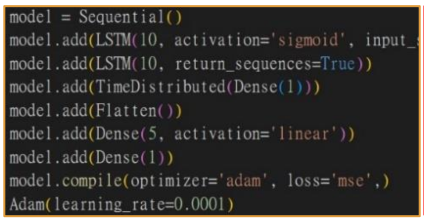
To fine-tune the model parameters to the optimal level and avoid overfitting (as right-side line chart) to a specific validation set, we employ cross-validation to ensure model generalization.

1. Split the data into training and testing sets, with the testing set reserved for final evaluation.
2. Further split the training set using K-fold Cross-validation.
3. Perform grid search with hyperparameters on each fold, training and evaluating the model K times.
4. Select the best set of hyperparameters based on the cross-validated performance and train on training set.

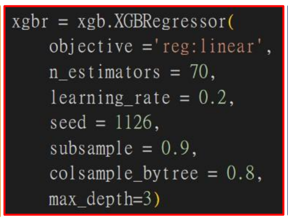
Finally, evaluate the final model on the reserved testing set to assess its performance.

During this process, we observed significant disruption in the model's performance on October 9th and 10th, which are national holidays. Since our prediction target does not include national holidays, we chose not to introduce a separate feature for them. Instead, we excluded these two days from the training dataset.

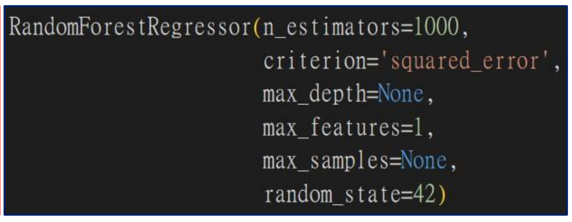
Below is our fine-tuning hyperparameters of each model (from left to right: LSTM, XGBoost, Random Forest).



```
model = Sequential()
model.add(LSTM(10, activation='sigmoid', input_
model.add(LSTM(10, return_sequences=True))
model.add(TimeDistributed(Dense(1)))
model.add(Flatten())
model.add(Dense(5, activation='linear'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse',)
Adam(learning_rate=0.0001)
```



```
xgbr = xgb.XGBRegressor(
    objective ='reg:linear',
    n_estimators = 70,
    learning_rate = 0.2,
    seed = 1126,
    subsample = 0.9,
    colsample_bytree = 0.8,
    max_depth=3)
```



```
RandomForestRegressor(n_estimators=1000,
                      criterion='squared_error',
                      max_depth=None,
                      max_features=1,
                      max_samples=None,
                      random_state=42)
```

Ensemble

We also attempted to enhance prediction results using an Ensemble approach. Initially, we employed Ensemble Voting to integrate gradient boost models from XGBoost, LightGBM, and CatBoost. Indeed, it proved beneficial for improving results, although the growth was exceptionally limited. The score increased from the initial 0.59 to around 0.60. This situation could also be attributed to the similarity in the methods of the sub-models used during the ensemble, potentially limiting the diversity and impact on overall performance. [4]
In addition, we have attempted to ensemble LSTM, gradient boosting and random forest, but encountered challenges during development, did not make it before the competition deadline. This might serve as future work for improvement.

Result

STAGE	**1st Public Score**	**1st Private Score**	**2nd Public Score**	**2nd Private Score**	**3rd Public Score**	**3rd Private Score**
Gradient Boosting (XGBoost)	0.3137	0.3404	0.3056	0.4344	0.3178	0.4656
Neural Network (LSTM, keras)	0.3943	19.523	0.4100	19.994	0.4000	19.640
Random Forest (scikit-learn)	0.5516	15.034	0.4173	19.946	0.3709	18.542

Comparison

From the results, it appears that both neural network and random forest have poor prediction scores on the private leaderboard, showing signs of overfitting. Only XGBoost manages to maintain similar prediction scores for both public and private leaderboards. This might be due to the strong time dependency of the data when handled by LSTM, coupled with the increased model complexity introduced by Keras.
Additionally, since the number of bicycles is influenced by various factors such as weather, holidays, final exams, crowd levels, and other "noise," introducing regularization in XGBoost leads to better generalization. In terms of structural characteristics, the decision trees in random forests are independent, making it less flexible compared to the gradient boosting algorithm used by XGBoost, which can adjust in each iteration. [5]
In summary, in the public and private testing phases across the three stages, XGBoost consistently performed the best. Therefore, our recommendation is to use the XGBoost model for predicting in this final project.

Pros and Cons

Regarding our recommended XGBoost model, we offer several observations on its strengths and weaknesses.

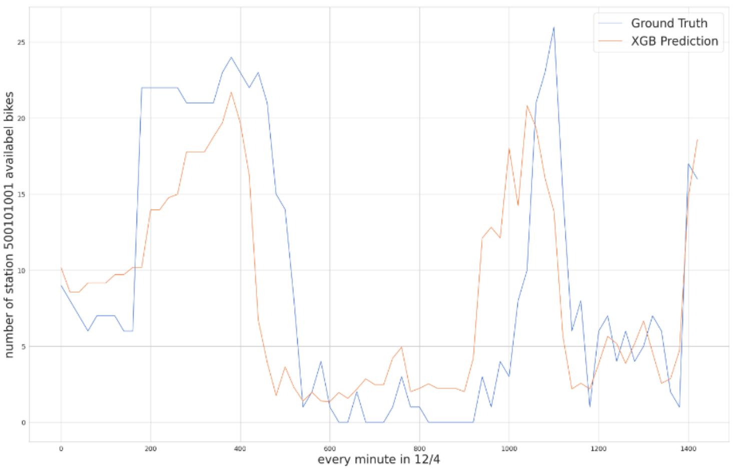
Advantages:

1. Fast training speed, facilitating parameter tuning.
2. Strong generalization, less prone to overfitting compared to other models.
3. Minimal need for standardization or extensive feature engineering due to its inherent capabilities.

Disadvantages:

1. In the presence of complex data, increasing tree depth leads to significantly longer computation times, consuming substantial CPU/GPU resources.
2. Parameter tuning often involves exhaustive grid search, which can be cumbersome.

Discussion



Finally, our observation of the YouBike quantity at different times revealed that the vehicle counts change rapidly during transitions in pedestrian flow. It can go from the maximum to zero or vice versa in less than 20 minutes. Predicting slightly faster or slower by 20-40 minutes can result in significant scoring errors. Additionally, the quantity between midnight and 8 a.m., although stable, is highly influenced by uncertain factors, contributing to many errors and losses. In future optimizations, consideration of this aspect will be crucial. (The figure illustrates the vehicle count at station 500101001 on 12/4 as an example.)

Work Loads

洪柏濤	Linear Regression, GBDT, and XGBoost Models Building and Testing
林東甫	EDA, Feature Engineering, LightGBM, Catboost, Neural Network, Fine-tuning, Ensemble
黃正渝	Data Preprocessing, Random Forest, LSTM and XGBoost Models Building and Testing

(PS. Some models, like Linear Regression, were excluded from the final report due to poor performance, while others, like GBDT, were excluded due to its relatively high similarity in model characteristics with XGBoost.)

Reference

[1] **Handling cyclical features, such as hours in a day, for machine learning pipelines with Python example:**
http://www.sefidian.com/2021/03/26/handling-cyclical-features-such-as-hours-in-a-day-for-machine-learning-pipelines-with-python-example/

[2] **K-Means Clustering in Python: A Practical Guide:**
https://realpython.com/k-means-clustering-python/

[3] **Mean Encoding – Machine Learning:**
https://www.geeksforgeeks.org/mean-encoding-machine-learning/

[4] **sklearn.ensemble.VotingClassifier:**
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

[5] **An introduction to XGBoost regression:**
https://www.kaggle.com/code/carlmcbrideellis/an-introduction-to-xgboost-regression