A. Introduction

The primary objective of this project is to predict the number of bikes in 112 stations around NTU for every 20-minute interval.We employ LSTM and CatBoost algorithms to develop a predictive model for Youbike demanding, forecasting future usage based on historical data.

The dataset includes station-specific information like station ID, name, district, latitude, longitude, and address. Additionally, it contains timestamped data in 24-hour format, featuring maximum station capacity, current bike count, available parking slots, and station availability status.

B. Data Pre-processing

I. Data Separating: First using pandas.read_json() import different station ID into python as dataframe by sno_test_set.txt ,and segment data set into one data per 20 minutes.

II. Missing Values : Using package pandas filled_data = data.fillna(method='ffill'), filled the missing data with the backward data, meanwhile, filling the last missing data with the forward data, making sure that NAN doesn't exist.

III. Features Architecture:

Weather：I have collected weather information from various stations, including rainfall and temperature. Since the information on the internet is compiled every hour, we have set the weather data every 20 minutes within an hour in our training data to be the same value.

Coordinate：Collected from a given dataset。

Distance：According to literature, the distance of the bike station to the nearest MPT. We collected the coordinates of the MRT station and calculated the nearest distance of the bike station to the nearest MPT, with a haversine formula.

Day of week：Determine which day of the week by the date of data set (0~6)

Minute of day ：Which minutes of the day(20, 40, 60…)

ID：ID of station

C. Machine Learning Approaches Explored

Approach 1 and Approach 2 share the same model

Long Short-Term Memory (LSTM), is a specialized type of neural network designed to overcome the limitations of traditional recurrent neural networks (RNNs). It excels in capturing long-term dependencies in sequential data by incorporating memory cells. At predicting future bike demand, which is based on past usage patterns considering factors like time, weather, or holiday, these features make LSTM networks particularly effective in modeling sequences.

Advantages:

- Handling Long-Term Dependencies: LSTM is designed to address long-term dependency issues, making it effective in understanding relationships in long sequences of data.

- Adjustable Parameters: It offers flexibility with various adjustable parameters, allowing adaptability to different data and tasks. (which can also be viewed as a kind of disadvantage)

Disadvantages:

- Complexity: LSTM's complexity requires tuning multiple parameters, leading to longer training times and increased computational demands.
- Overfitting: It may overfit, especially with small datasets or overly complex models, impacting generalization to new data. (And that's what we may concentrate on later)

Approach 1

Additional Data Preprocessing:

Normalization: we use the calculating method of $(maximum - minimum)/mean$ to transform the column data into 0-1, improves training efficiency and model performance.

Input Data: three-dimensional Array: including time steps and features. This structure is typical for LSTM models dealing with time series data.

Three LSTM Layers: the model consists of three LSTM layers, each with 256 neurons.

Dropout Layers: following each LSTM layer, there is a Dropout layer with a dropout rate of 0.01. The dropout layers help in reducing overfitting by randomly omitting a fraction of the features during training.

Dense Output Layer: the final layer is a Dense layer that outputs a single value, representing the prediction result.

Compilation:

Loss Function: Mean Squared Error (MSE) as its loss function, a standard loss function for regression problems and measures the average squared difference between the estimated values and the actual value.

Optimizer: Adam optimizer is used for training. which combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

	Parameter1	**Parameter2**	**Parameter3**
Dropout Rate	0.1/ 0.1/ 0.1	0.01/ 0.01/ 0.01	0.1/ 0.01/ 0.1
Optimizer	layer = 128 neurons	layer = 128 neurons	layer = 256 neurons
Error Validation	0.49743	0.48396	0.45506

	Method 1	**Method 2**
Extra Features	datetime/ tot/ act/ minutes of day/ dayofweek/ temperature/ rain/ holiday	datetime/ tot/ act/ minutes of day/ dayofweek/ temperature/ rain/ holiday/ station id

DataSet	Every station data per day in same weekday with eight features	All station data per day in same weekday with nine features
Concept	Train different station ID in different weekday separately [One model/ station, weekday]	Train all station in different weekday at the meantime (in same model training) [One model/ weekday]
LSTM Layer.	Three LSTM layers, each with 256 neurons.	Three LSTM layers, each with 256 neurons.
Dropout Layer	parameter we set	parameter we set
Dense Output Layer	Dense(1)	Dense(1)
Optimizer	Adam	Adam
epoch	300	300

Approach 2

Additional Data Preprocessing:

After processing the null value, I extract data every 20 minutes every day and station to construct a dataframe. Then I take the first 1008 rows of the dataframe and separate them from the middle. Take the first half as the first x_train (504rows means each of my x_train contains data every 20 minutes of the whole week) and the second half as the first y_train. After that, I discard the first 4 rows of the dataframe and do the same thing again. Thus, I get 504 rows X 6 features every 80 minutes as my x_train and also my y_train.

One LSTM Layers: The model consists of one LSTM layer, with 256 neurons.

Dropout Layers: Following with the LSTM layer, there is a Dropout layer with a dropout rate of 0.3.

Dense Output Layer: The final layer is a Dense layer that outputs a single value, representing the prediction result.

Input Data: Three-dimensional Array: including time steps and features. This structure is typical for LSTM models dealing with time series data.

Compilation is the same as approach 1.

Parameters:

	Parameter1	**Parameter2**	**Parameter3**
Dropout	0.01	0.1	0.3
Optimizer	adam	Sgd	rmsprop
LSTM units	16	128	256

We try out all 27 combinations of three parameters.

Result:

The best results are the combination of Dropout = 0.3, Optimizer = adam, LSTM units = 256 with error of 0.73.

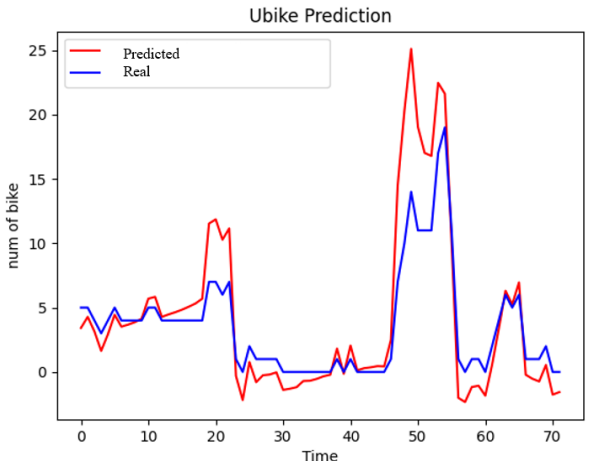

Figure 1. with Approach 1
The example results of one station 12/04

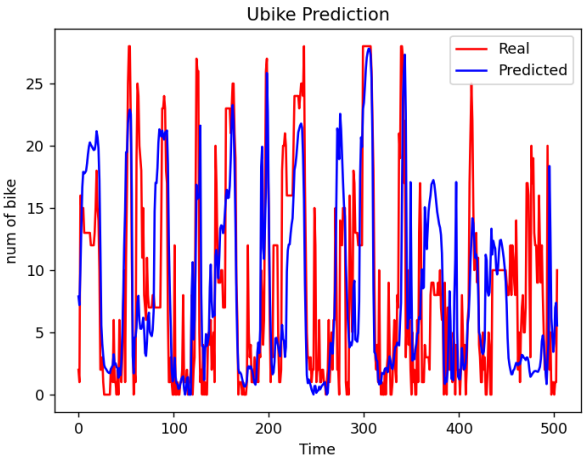

Figure 2. with Approach 2
The example results of one station 12/18 to 12/24.

Approach 3

CatBoost is an algorithm based on gradient boost. It's known for handling categorical data effectively without the need for extensive preprocessing, and it provides excellent out-of-the-box support for model interpretation and evaluation. CatBoost is widely used for its speed and accuracy in a variety of tasks, from regression to classification, and is well-suited for both small and large datasets. It also provides lots of visualization tools, so we can analyze the process of training model and prediction more intuitively.

Data split:

using train_test_split() from sklearn.model_selection to split the data into x and y, split the data into 80% of training data and 20% of test data.

```
# split data into train data and test data
x_train, x_test, y_train, y_test = train_test_split( *arrays: data_frame.drop( labels: ["sbi"], axis=1), data_frame["sbi"],
                                                    test_size=0.2, random_state=42)  # 80% train 20% test
```

Build model

I built a model using CatBoost's CatBoostRegressor() and compared various parameters using grid_search_result(). I chose to test iteration, learning rate, depth, and border_count, and identified the best combination

According to the result of grid search, the best combination is shown as below:

```
{'border_count': 70, 'depth': 10, 'l2_leaf_reg': 1, 'iterations': 4000, 'learning_rate': 0.2}
```

Since the iteration tended to stabilize around 1500, and the learning rate was within the boundaries I set, I modified the parameter range and retrained the model. This led to a new optimal combination, reducing the error from 0.43 to 0.41.

Figure 3 : New training grid

```
# decide the best parameters with grid
grid = {
    'iterations': [500, 1000, 1500],
    'learning_rate': [0.1, 0.05, 0.01],
    'depth': [2, 4, 6],
    'l2_leaf_reg': [1],
    'border_count': [30, 40, 50]}
```

Figure 4 : New best combination

```
{'border_count': 50, 'depth': 6, 'l2_leaf_reg': 1, 'iterations': 1500, 'learning_rate': 0.1}
```

prediction results:

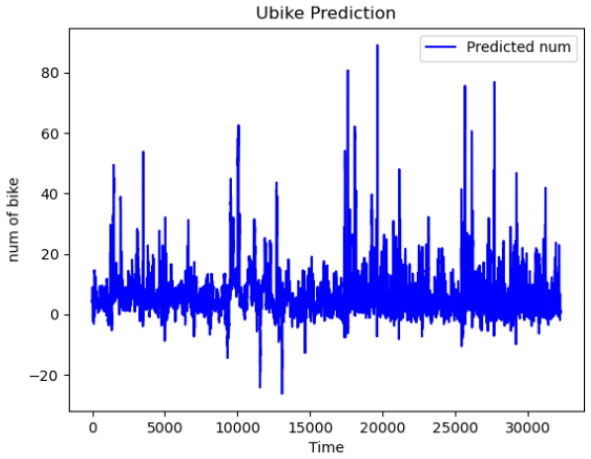

Figure 5 : The prediction of current bike account

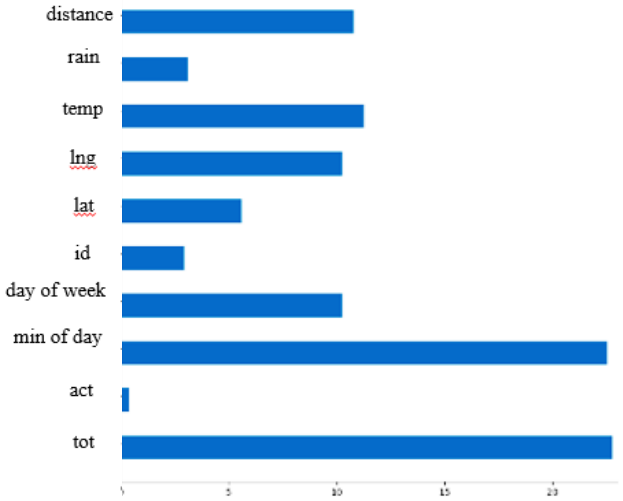

Figure 6 : The importance of each features

Based on our results, we observed in Figure 5 that among the features we added, the distance to the nearest MRT station and temperature both have a significant impact, while the importance of rainfall is relatively low. We speculate this is because there are too few rainy days, with most of the data being nearly zero, hence it does not significantly influence the prediction results.

In the loss over iteration graph (figure 7),
not only does Ein tend to stabilize after 1000 iterations, but Eval also shows a significant decrease, indicating that our model is reproducible.

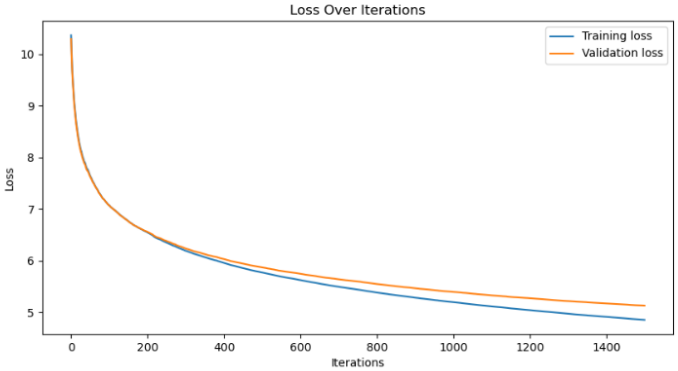

Figure 7 : loss over iteration

D. Comparison of Approaches

Efficiency:

In terms of training period, catboost has the highest efficiency running with 1 minute per station for public tests, yet LSTM has the lowest with 40 minutes per station. Other than that, the pre-process time we need in catboost comes much lower than LSTM since we need only one dataframe as an input for catboost.

Scalability:

In our LSTM model (approach 2), we found out that our model performance worsens when we scale up more input data size. We get 0.735 in the public test when we use the data only in October, yet we get 1.06 when using data in October, November and December. In comparison, for the catboost model, it shows better performance with larger size of dataframe by extracting the data per 60 minutes to 20 minutes. The error went from 0.45 to 0.44

Interpretability:

Catboost definitely has better interpretability than LSTM since it provides a lot of visualization tools such as importance of features, error over iteration and shape etc. So that we can make it easier to analyze and comprehend the result compared to LSTM.

Reproducibility:

In terms of reproducibility of data, we can collect extra data from the opening source and also update the training data easily so that we can retrain the model in a more efficient and handy way.

As for reproducibility of model, good reproducibility of model means close Ein and Eout so that we can apply the model under any circumstances. Considering the result from kaggle competition, it turns out that the difference between public test error and private test error of Catboost (0.01) is smaller than LSTM (0.07). Last, we include a model saving program so that it becomes unnecessary to retrain the model over again every time

E. Recommendation

Concluding the preceding information, after the comparison between Catboost and LSTM (approach1 & 2), we recommend Catboost! The reasons are as follows:

1. Better efficiency
2. Easier to analyze and interpret
3. Lower Error
4. Easier in data pre-process

F. Reference

1. CODiS CWA Observation Data Inquire Survice
   https://codis.cwa.gov.tw/StationData0
2. Journal of Photogrammetry and Remote Sensing Volume 24, No.4, 2019, pp. 245-255
3. Application of LSTM predicting Youbike needing
   Enterprisehttp://ielab.ie.nthu.edu.tw/110_IIE_project/3/pdf_word/110034540%E5%BE%90%E9%98%A1%E7%8F%8A.pdf
4. northen-taiwan-metro-station
   https://github.com/repeat/northern-taiwan-metro-stations