**NYC Citi Bike Demand Location Prediction and Analysis**

**Group 6-Project 14**

Yandong Luo, Stanley H. Gu, Jingwei Bao, Weichen Liu

**Introduction**

The booming bike-sharing programs in New York City have grown in ridership, bikes, and bike dock stations ever since 2013. With the rising demand for citi bikes, one of the main challenges that citi bikes confront is their dependence on trucks to redistribute the bikes across stations. If the bikes were not redistributed correctly, dock stations may become completely full or completely empty – it is rendered unusable. Hence, predicting demand for bikes is important to Citi Bike’s company to both reduce operating costs and increase ridership.

To avoid poor customer experience and loss of revenue due to under-utilized bikes, this paper aims to determine where exactly more Citi Bikes should be placed at what times of the day.

We aim to first explore the relevant features that influence the natural aggregated flow of customers' destination trips throughout the city, and process these features to adapt the model that predicts the bike loss in each station at a given time. Finally, the predicted bike loss outcomes will be compared with the actual data to evaluate their usefulness.

**Data Exploration**

The NYC Citi Bike Rentals’ official website provides its bikes’ system-data with the features of*Start Station name, End station name, Date, Start time, End time, Trip Duration* *(seconds), Start latitude/longitude, End latitude/longitude, and User Type (customer or member), Gender, Year of Birth.*

Additionally, this paper further considers weather & holiday as external features that may influence the bike loss in each station at a given time.

**Data Processing & Selection**

Upon careful evaluation of the given data, certain features are dropped for the model evaluation. These features are dropped either due to the lack of data over the years of 2019-2022(*Gender, Year of Birth*), or the irreverence and adaptive issue of the data to the bike loss in each station at a given time(*Trip Duration, Start latitude/longitude, End latitude/longitude, User Type*). Therefore, we performed data cleaning on the original data and filtered missing values ​​to ensure the smooth operation of the model.However, the features of *Trip Duration, Start latitude/longitude*, and *End latitude/longitude* are still used for data visualization for a better understanding of the data.

The *Start/End Station name features* are converted to station id: a categorical variable from a range of 0 to 95, whereas the *Date* features are converted to *day of weeks*, which are categorical variables ranging from 0-6(Mon-Sun).

Additionally, instead of having 24 hours as the categorical values of the *Start/End time* features, the *Start/End time* is merged into 4 time slots per day (6 hours per time slot) in hope of reducing the complexity of the model and making it easier to converge. As a result, two new features are introduced: *bike in and bike out*. The *bike in* and *bike out* are calculated by summing up the numbers of *the end station name* and *start station name* for each trip(at a given station and time slot of a day). Bike loss, the predicting variable, can then be defined as the *bike in-bike out* per station at a given time slot.

Precipitation\_Snow depth\_Snowfall, and Average Temperature are further breakdown from the *weather* feature for data processing, and the *holiday* feature is stated as a binary categorical feature, where 0 is the non-holiday date and 1 is the holiday date.Hence, the feature and predictor variables are as follows:

**Features: Predictor/Label:**

Dayofweek Bike\_loss

Station\_name(Station\_id)

Bike\_in

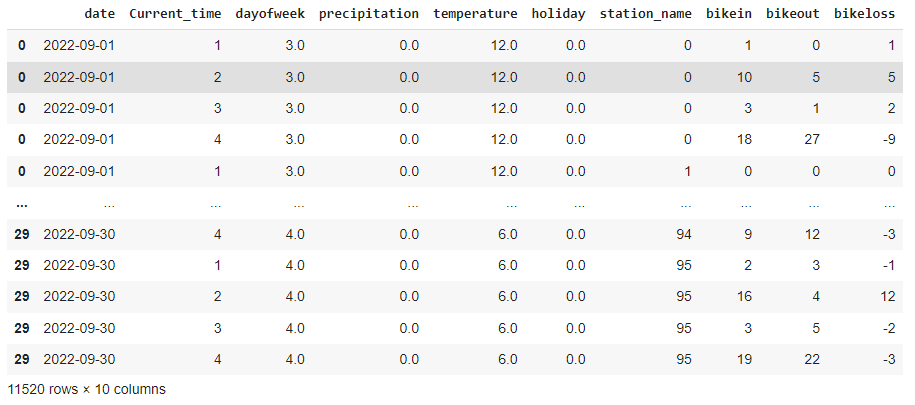
Bike\_out

Current\_time (Timeslot)

Precipitation (Precipitation\_Snow depth\_Snowfall)

Temperature (Average Temperature)

Holiday

****Table 1. Processed data

**Data Visualization & Analysis**

We created a map showing all start stations which gave us an intuitive thought about how those stations are distributed.

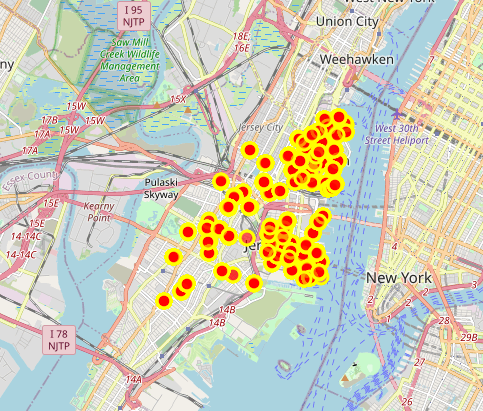
****

Figure 1. End stations viewing on the map

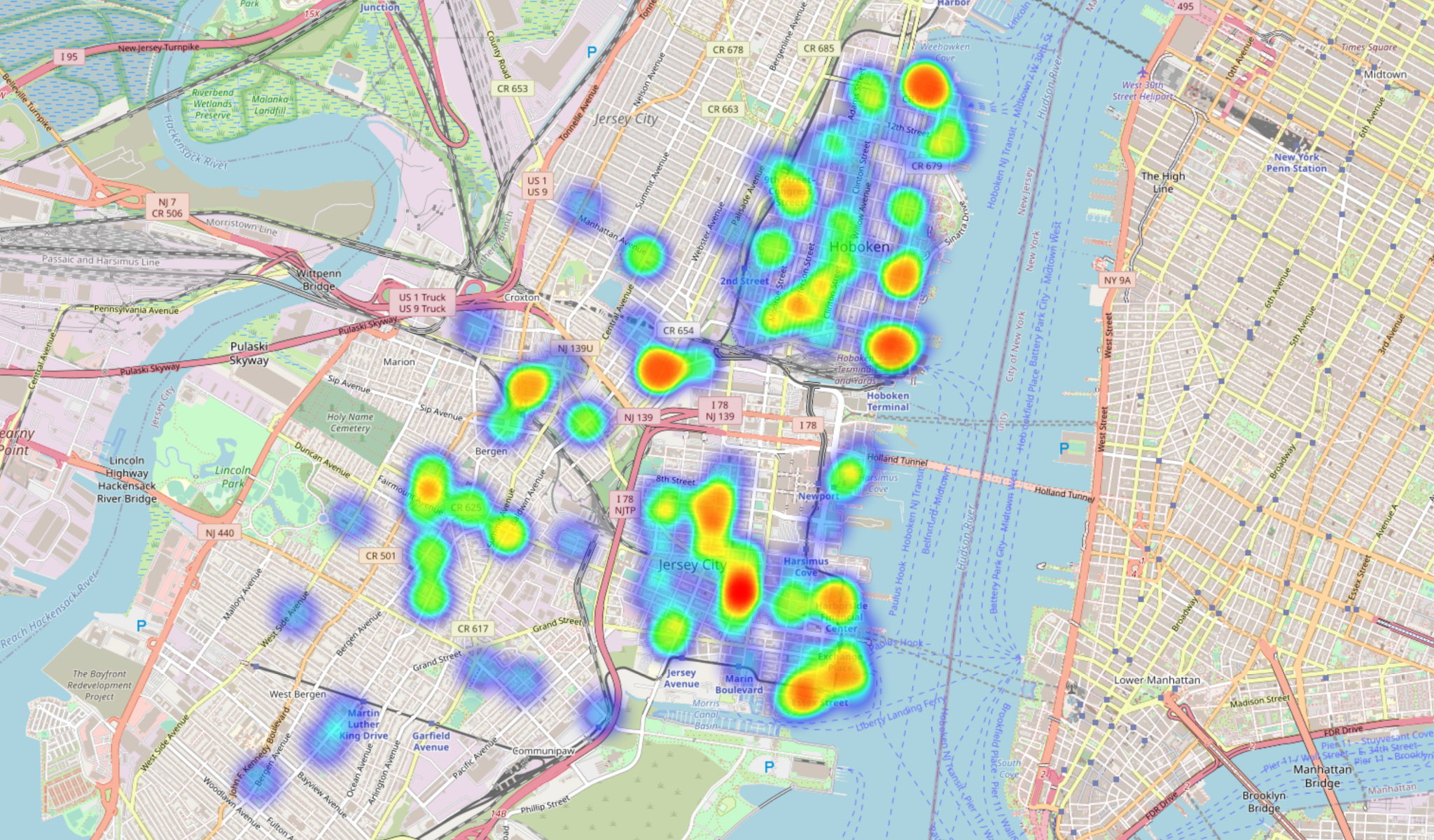


Figure 2: Figure 1. End stations viewing heat map

We used a histogram to visualize the number of people at each station in one day as shown in Figure 1. Each station was represented from 0 to 155.

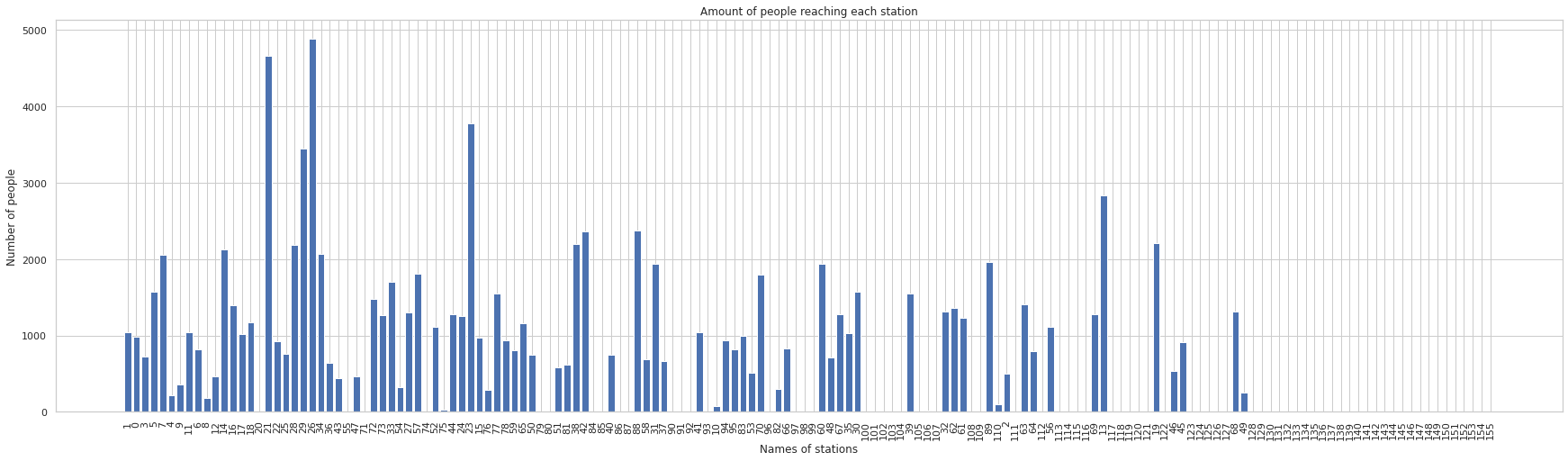


Figure 3. The number of people reaching each station.

We assume that there is a linear relationship between the destination and the starting point, the type of bicycle, the precipitation of the day, bike in, bike out, the temperature, the timeslot, and the driving time. So, we use the Pearson correlation coefficient to examine the strength and direction of the linear relationship between continuous variables as shown in Figure 1. It is easy to know that current time has a more obvious relationship with bike in and bike out. At the same time, there is also a proportional relationship between bike in and bike out.

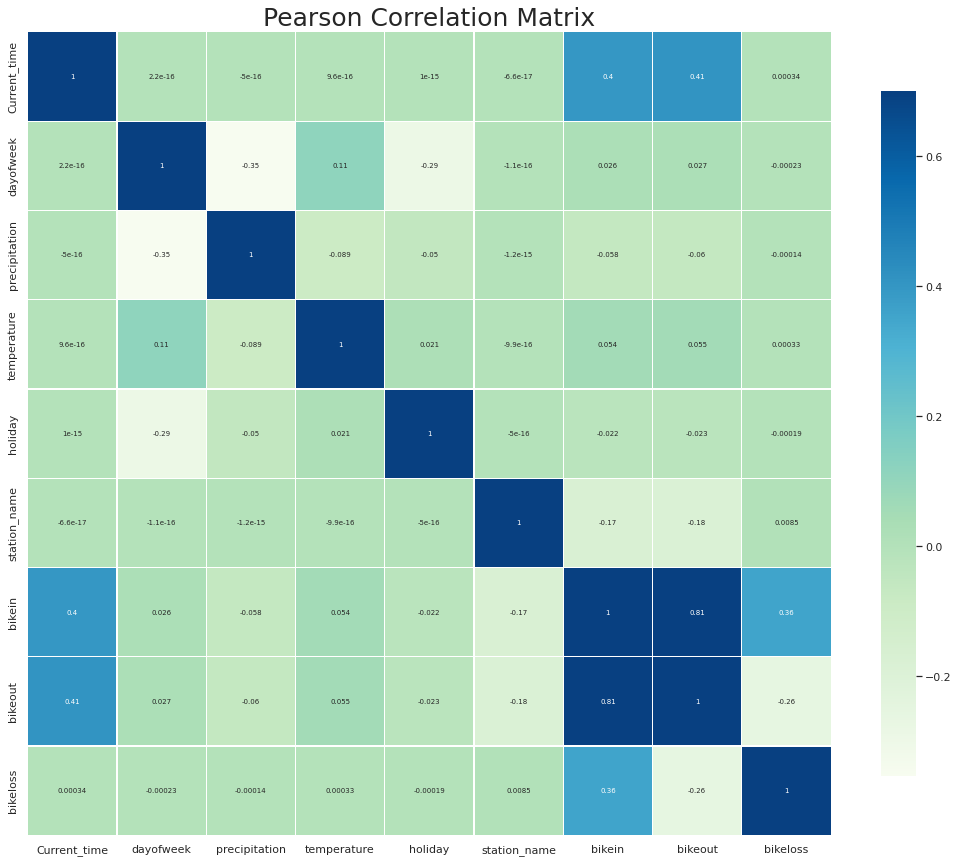


Figure 4. The Pearson correlation.

**Model Analysis & Accuracy Function**

The prediction model is a Multi-Layer Perceptron (MLP) constructed with an input layer that contains the 8 features, three hidden layers, and one output layer that predicts the bike\_loss. Among the hidden layers, Relu functions are applied to prevent the neurons from going too low or too high. Xavier initialization is also used as the initialization scheme for this neural network model.

We then further investigated the effects of mini-batch learning and different optimizers. We selected mini-batch sizes of 4, 50, and 1000, and optimizers of RMSprop, SGD, and Adam for Investigation. Each were run with 20 epochs.

| batchsize\optimizers | Adam | SGD | RMSprop |
| --- | --- | --- | --- |
| 4 | Accuracy = 0.9951  Runtime : 2min | Accuracy = 0.9993  Runtime : 1min | Accuracy = 0.9895  Runtime : 2min |
| 50 | Accuracy = 0.9900  Runtime : 2min | Accuracy = 0.9816  Runtime : 1min | Accuracy = 0.9954  Runtime : 2min |
| 1000 | Accuracy = 0.9557  Runtime : 2min | Accuracy = 0.9979  Runtime : 1min | Accuracy = 0.9971  Runtime : 2min |

Table 2. Mini-batch size & optimizer investigation

It can be seen that there is a general decrease in accuracy as batch size increases for the Adam optimizer, this is reasonable since batch size controls the accuracy of the estimate of the error gradient when training neural networks -When using a smaller batch size, calculation of the error has more noise than when we use a larger batch size. This can help the model to jump out of the local minimum and hopefully find a better one.

The SGD optimizer has a rather oscillating accuracy among the three models with different batch sizes. This phenomenon may be due to the nature of the SGD optimizer. SGD can not truly find the minimum, it will keep oscillating around the minimum point and can go on indefinitely. Such oscillation will result in the oscillating accuracy among the three models with different batch sizes.

The RMSprop optimizer, on the other hand, increases the accuracy as batch sizes increase. We believe this unusual and counterintuitive phenomenon is because RMSprop introduces the mini-batch without splitting the gradient relationship between each mini-batch set.

Besides the investigation of the effects of mini-batch learning and different optimizers. We also tuned hyperparameters (training: testing) to ⅛:⅞, ⅓:⅔, and ⅔:⅓ of the processed data. Among these tuned hyperparameters, we observed that ⅔:⅓ yields the result with the highest accuracy.

To sum up, while the SGD optimizer with a 1000 batch size gives higher accuracy and faster running time, its accuracy is relatively unstable as it keeps oscillating around the minimum point. Adam optimizer, on the other hand, has a longer runtime under the same epoch, yet yields the lowest accuracy. Hence, we chose RMSprop as our optimizer with a batch size of 10, which yields an accuracy of 0.9961.

**Conclusion**

This study predicts the bike shortage pattern of New York City’s Citi Bike system for 4 evenly divided time slots of each day in each station. Besides the factors provided by the Citi Bike’s official website, we used weather, holiday as external factors to further predict the bike shortage. We also combined the start\_station & end\_station to bike\_out and bike\_in to reduce the complexity of the model and allow it to converge with ease. As a result, our model predicted a 0.9961 accuracy compared with the actual bike loss.

However, there are still improvements that can be made. While the MLP model yields a high accuracy for this citi bike problem, the GNN model is a more promising model when dealing with problems like this. In the future, we would like to construct a GNN model and compare its performance with the current MLP model.