Modeling under uncertainty Final Project

Negar Akbarzadeh (na332)
Rojin Zandi (rz285)
Jia Zhao (jz538)

November 27, 2021

1 Introduction

In this project we are working on NYC Bike data which contains 3084537 data points and each data point is a ride that happened in July 2021. Some of the important features of this dataset are started at(when the ride started), ended at(when the ride ended), start station ID, end station ID, member or casual.

All code details and be found in our submitted code.

2 Warmup Questions

1. Using the start time and end time, compute the duration of each ride in minutes and plot the histogram of ride durations.

   Solution: In order to find the duration of each ride, we compute the difference between start time and end time of each ride and convert it to minutes. Then the histogram plot can be plotted by matplotlib library. Here is the ride duration plot:

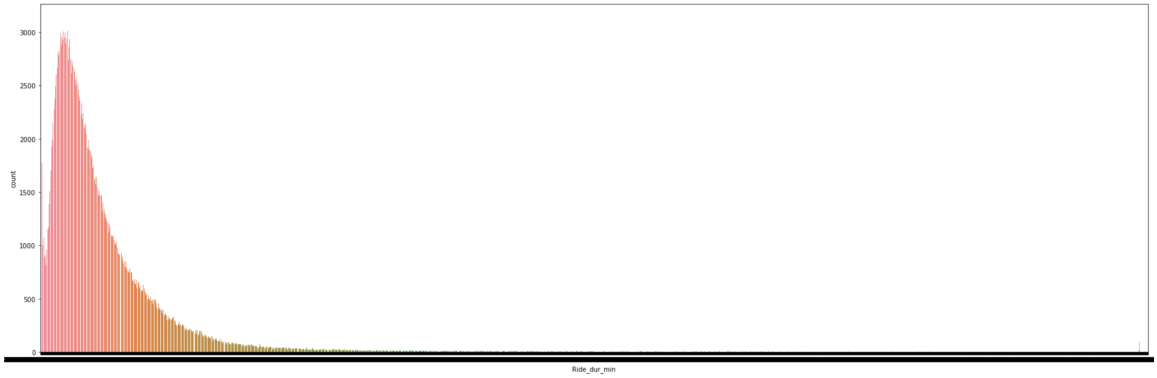

Figure 1: Histogram of ride duration in minute

   NOTE: We can see that there are "outliers" whose ride duration are super high. We keep those data in the warm up question and filter them later in the project. This decision may affect our results for the following warm up questions.

2. What is the expected ride duration (i.e., the average ride duration)? What is the empirical variance of ride duration? What is the probability that a ride duration is greater than 20 min?

   Solution: Using .describe() command in python, we can get statistical information of ride duration. The average of ride duration is 18.02 (minutes).
   The standard deviation of ride duration is 83.16 and the variance is 6915.028.
   To compute this probability, we count the number of rides which are longer than 20 minutes

and divide it by number of all rides and the result is 24.97%.

3. What is the probability that a ride duration is greater than 20 min conditioning on the fact that the user is a CitiBike member? Note that the last column gives whether the ride is for a casual client or a CitiBike member.

   Solution: First we must find number of rides that have been done by Citybike members, and then find how many of these rides lasted longer than 20 minutes. Finally, divide them and the result of this conditional probability is 19.42%.

4. Suppose that the duration of some ride is more than 25min. What is the probability that this ride belongs to a CitiBike member?

   Solution: First step is to count number of rides that lasted longer than 25 minutes and the cont how many of thos have been done by members. The probability of a ride longer than 25 minutes given that the rider is a member is 48.28%.

3 Project

In this part our goal is to calculate the stationary distribution of three most popular stations during weekdays.

3.1 Preprocessing

We load the dataset as pandas.dataframe.

We first clean the data: delete the weekends, split each day to two blocks (morning and afternoon) and divide each block to periods of 10 minutes.

To do so, we add three columns 'start hour', 'start min' and 'weekday' based on 'started_at':

```
# identify the start hour and weekday

df['start hour'] = df['started_at'].dt.hour.astype('int')
df['start min'] = df['started_at'].dt.minute.astype('int')
df['weekday'] = df['started_at'].dt.weekday.astype('category')
```

Figure 2: Find the start hour, start minute and weekday of each ride

We assign 0 to 6 as value of 'weekday' for each data point, where 0 to 4 indicate weekdays and 5 to 6 indicate weekends.

```
# delete weekends

indexNames = df[df['weekday'].isin([5, 6])].index
df.drop(indexNames, inplace=True)
```

Figure 3: Delete weekends

The next step is splitting each 24 hour to two blocks of morning and afternoon, where morning is 1 and afternoon is 0:

```
# use start hour to represent the block of the day
# morning -> 1 and afternoon -> 0

df['mor/aft'] = np.where(df['start hour'] < 12, 1, 0)
```

Figure 4: Split days to two blocks

Then, divide each block (morning or afternoon) to a period of ten minutes.

```
# split morning/afternoon into blocks based on 10 min

df['period'] = df.apply(lambda row: row['start hour'] % 12 * 6 + row['start min']//10 + 1, axis=1)
```

Figure 5: Divide each block to periods of 10 minutes

We also delete all the rides with ride duration larger than two standard deviation above the mean ('outliers' of the ride duration):

Figure 6: Delete Outliers of Ride Duration

We now find the three most popular stations for **weekdays** and their capacity from citibike online sources. The most popular stations are:

- West St & Chambers St (ID:5329.03) with 29 docks (labeled as 2 in our code)
- E 17 St & Broadway (ID:5980.07) with 66 docks (labeled as 1 in our code)
- W 21 St & 6 Ave (ID: 6140.05) with 50 docks (labeled as 0 in our code)

Since we only consider rides started (or ended) at the above three stations, we delete data points neither started nor ended at the above three stations:

```
# keep only the top 3 stations
df = df[ ((df['start_station_id'] == 0) | (df['end_station_id'] == 0))
                | ((df['start_station_id'] == 1) | (df['end_station_id'] == 1))
                | ((df['start_station_id'] == 2) | (df['end_station_id'] == 2)) ]

```

Figure 7: Delete data points with other stations

3.2 Markov Chain

We simulate one Markov Chain (MC) for each station in each time block which adds up to 6 Markov chains with 6 different transition probabilities. The states in our MC are number of available bikes in the station. For example, 'West St & Chambers St' station has 29 docks which results in 30 different states, from 0 to 30 available bikes.

The MC for each station is an $n \times n$ matrix, where n is number of docks + 1. In order to compute the transition probability of each MC, we first find the frequencies. Let f be the frequency matrix and f_{ij} denote the frequency of starting at state i (i bikes) and end at state j (j bikes) in one period. The transition probability $p_{ij} = f_{ij} / \sum_{j=0}^{n} f_{ij}$ (if $\sum_{j=0}^{n} f_{ij} > 0$).

3.2.1 Compute transition frequency

We compute the frequencies as follows. The function returns the frequency matrix based on time block (morning/afternoon), station id, number of bikes at the start and the capacity at the station.

```
def LB(MA, id, numOfBikes, capacity):
    # MA: morning (1) or afternoon (0)
    # id: station id 0, 1, 2
    freq_matrix = np.zeros([capacity+1, capacity+1])

    pre_bikes = numOfBikes
    for i in range(72):
        ave_leaving = df[ (df['period'] == i) & (df['start_station_id'] == id) & (df['mor/aft'] == MA) ].shape[0] // 22
        ave_arriving = df[ (df['period' ] == i) & (df['end_station_id'] == id) & (df['mor/aft'] == MA) ].shape[0] // 22
        cur_bikes = pre_bikes + ave_arriving - ave_leaving

        if cur_bikes < 0:
            cur_bikes = 0
        elif cur_bikes > capacity:
            cur_bikes = capacity

        freq_matrix[pre_bikes][cur_bikes] += 1
        pre_bikes = cur_bikes
        # print("Period %d -> %d left, %d arrived, %d available at %d." % (i, ave_leaving, ave_arriving, cur_bikes, id))

    return freq_matrix
```

Figure 8: Compute the frequency matrix

Considering that the data contain the information of 22 days (total number of weekdays in July 2021), we divide the number of bikes by 22 to obtain the average number of bikes arriving at / leaving from one station in one period.

We found that the patterns for each station in the morning (and afternoon) are different. So we use the parameter `numOfBikes` to decide the number of bikes available at each station at the start of morning (and afternoon).

For each period, `pre_bikes` is the number of bikes at the end of last period (at the start of this period) and `cur_bikes` is the number of bikes after this period. If too many bikes arrive at the station and the number of bikes available goes beyond `capacity`, we record `cur_bikes` as `capacity`; similarly, if too many bikes leave from the station when there is no bike available, we record `cur_bikes` as 0. This is simply because the number of bikes cannot be larger than the number of docks or negative (smaller than 0). Our assumption is that there is always a citiBike staff at each station removing bikes more than the capacity to keep the station just full and providing just enough bikes for people to use when the station is empty.

For each period, the number of bikes transit from `pre_bikes` (state `pre_bikes`) to `cur_bikes` (state `cur_bikes`). To count the frequency of this transition, we increment one for `freq_matrix[pre_bikes][cur_bikes]`.

3.2.2 Covert transition frequency to transition probability

If f_{ij} is the transition frequency from state i to state j, the transition probability $p_{ij} = f_{ij} / \sum_{j=0}^{n} f_{ij}$ (if $\sum_{j=0}^{n} f_{ij} > 0$). We compute transition probability as follows. The function takes frequency matrix as input and output the corresponding transition probabilities (the Markov Chain).

```
def freq2prob(freq_matrix):
    # convert frequencies in each matrix to transition probabilities
    re = np.zeros(freq_matrix.shape)
    for i, row in enumerate(freq_matrix):
        if sum(row) > 0:
            for j, col in enumerate(row):
                re[i][j] = col / sum(row)

    return re
```

Figure 9: Convert frequency matrix to transition probability matrix.

3.2.3 Compute Stationary Probability

The last step in the project is computing stationary distribution for each MC. We use repeated matrix multiplication to approach the stationary distribution. There are also other methods such as left eigen-vector and Monte Carlo method.

The logic behind the repeated matrix multiplication is multiplying the MC by itself many many times, then each row converges to the stationary probability:

```
1  def staionary_prob(transition_prob):
2      pmn = np.linalg.matrix_power(transition_prob, 1000000)
3
4      # every row converges to the stationary probability
5      pi = pmn[0]
6
7      return pi
8
```

Figure 10: Compute Stationary distribution by repeated matrix multiplication

We multiply each MC by itself 1000000 times and get the stationary probability for it. Each stationary distribution is an $n \times 1$ vector, and the sum of all values in the vector must be equal to 1.

In order to prove the accuracy of our work, we show the sum of stationary distribution values. We use the following code to show our result:

```
for i, each in enumerate(sp):
    # print("Stationary prob for %d is:" %i, each)
    print("The non-zero stationary probability for %d is:" %i)
    for j, p in enumerate(each):
        if round(p, 4) > 0:
            print("\tstate %d with prob %.5f" % (j, p))
    print("Sum of stationary prob: %.2f" % sum(each))
    print('\n')
```

In the final result (11), we only output the non-zero stationary probability. The complete result can be found in our submitted code.

RECALL: we manually decided the start number of bikes for each station in each block as shown in the above table(1). This decision may affect our result of stationary distribution.

MC number	Station id	Station	Mor(1)/Aft(0)	capacity	# of Bike at Start
0	0	W 21 St & 6 Ave	0	50	50
1	0	W 21 St & 6 Ave	1	50	50
2	1	E 17 St & Broadway	0	66	66
3	1	E 17 St & Broadway	1	66	0
4	2	West St & Chambers St	0	29	29
5	2	West St & Chambers St	1	29	0

Table 1: Information for Each Markov Chain

```
The non-zero stationary probability for 0 is:
    state 0 with prob 1.00000
Sum of stationary prob: 1.00


The non-zero stationary probability for 1 is:
    state 0 with prob 1.00000
Sum of stationary prob: 1.00


The non-zero stationary probability for 2 is:
    state 12 with prob 1.00000
Sum of stationary prob: 1.00


The non-zero stationary probability for 3 is:
    state 42 with prob 0.57143
    state 43 with prob 0.14286
    state 44 with prob 0.28571
Sum of stationary prob: 1.00


The non-zero stationary probability for 4 is:
    state 0 with prob 0.93750
    state 1 with prob 0.06250
Sum of stationary prob: 1.00


The non-zero stationary probability for 5 is:
    state 0 with prob 0.72222
    state 1 with prob 0.04167
    state 2 with prob 0.01389
    state 3 with prob 0.04167
    state 4 with prob 0.02778
    state 5 with prob 0.08333
    state 6 with prob 0.06944
Sum of stationary prob: 1.00
```

Figure 11: Result for Stationary Probability