Final Project - Predicting Bike Departure in Chicago

Kefan Long; Huilin He

Dec. 19, 2017

1. Purpose of our prediction

Riding bike is an affordable and environmental-friendly approach to get around. As a bike share system in Chicago, Divvy is operating 580 bike stations in the city. However, there is rooms for improvement.

Current divvy bikes can be used by the "TRANSIT" app. When you open the map and click on different stations,a dialog will pop up, telling you how many bikes and available docks there are in certain point of time. Current functions are helpful for most of the trips, but they can be improved. Some popular stations experience high-speed circulation and the information provided only in real-time is not enough. It can be possible that when the app showed you that there were available bikes or docks 5 minute ago, you may arrive and find no bikes or docks.

Beside, some stations have long-time "net departure" or long-time "net arriving", which could last for hours or even days. For these stations, they will have either no enough bikes to use or no enough docks for riders to park their bikes in a long time. Users may simply regard these as "never have a bike" or "never have a dock" and never come to these stations, which is not we expect to see.

Therefore, we need to create a product to help improve these potential problems. If there is a new function which informs the predicted number of divvy trips within certain stations in a certain point of time, cylists will have more options, They can choose either to wait a shorter time to get a bike in a further station or walk the shortest distance to get the bike in a longer time. Besides, they will have the chance to find bikes or docks on these "never have a bike" stations or park their bikes in "never have a dock", only if the new function can provide a robust prediction. We firmly believe that with this new function, Divvy will provide more user-friendly and smarter usages for bike users, and promote the usage of this bike share system.

2. Data preparation - Setup

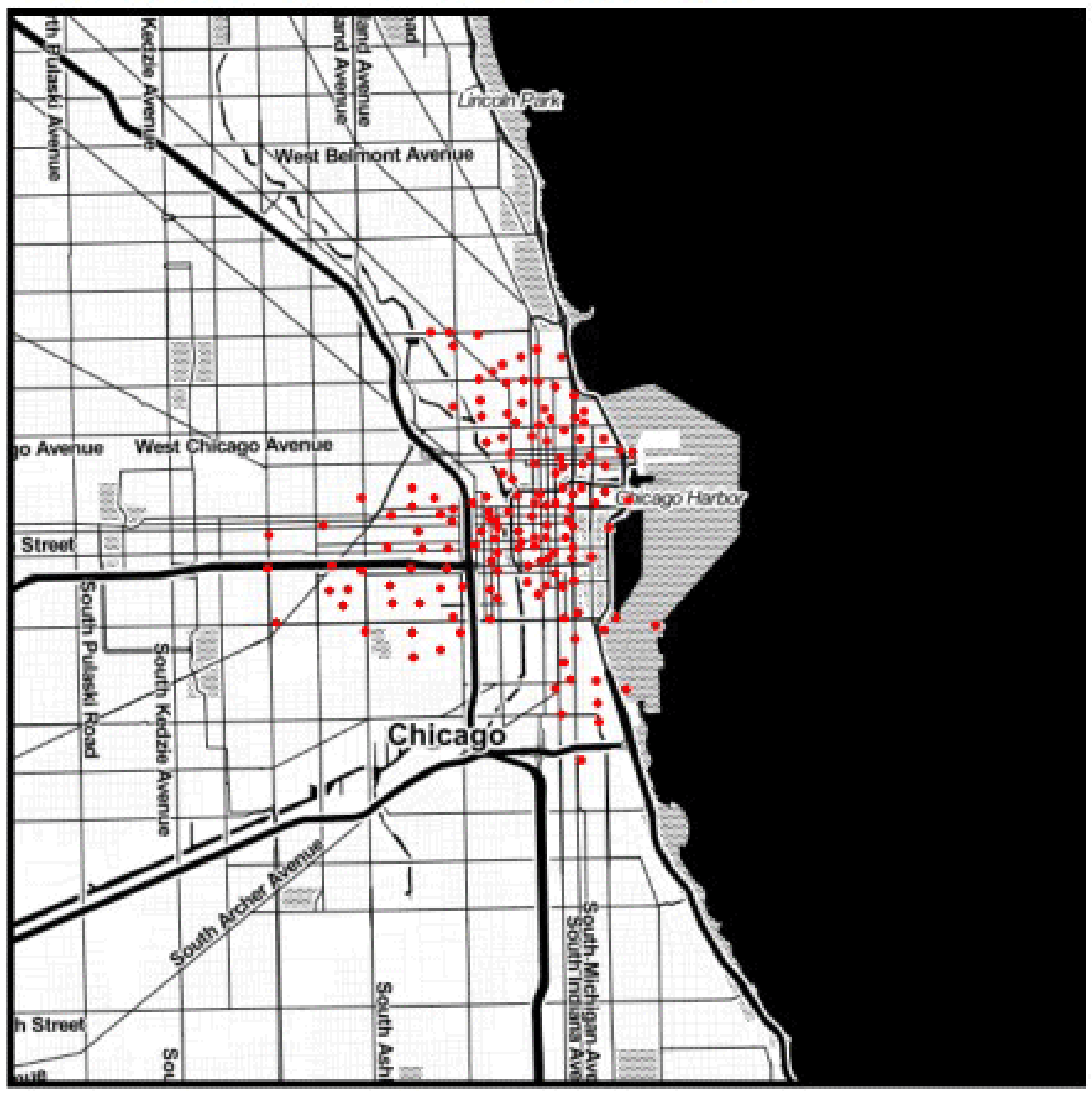
Below are the packages that we will use for the demo. We'll also include a mapTheme() function to plot the station and our prediction. To deal with information from data, we use *"lubridate"* package from R.

The first is to create a basemap displaying the target station we dicide to choose. There are 580 bike stations in Chicago in services. To reduce the amount of computation and keep the prediction as reprensentative, our prediction select stations around LOOP area, where there are higher possibilities of shortage in bikes or docks. The bike stations we select are located in four communities around the center city including Loop, Near West Side, Near South Side, and Near North Side. And the number of stations we chose is 144.

For each of the station, we also find basic information from different data sources, which include spatial variables and socio-economic variables. We'll list all of them in the third part.

After selection and exporting data from GIS, we get basic information of selected stations and can plot the location of the stations in R using their *longitude* and *latitude* from the attribute table.

Selected DIVVY Stations in Chicago



By selecting bike stations in different types of communities we can increase the variety of our samples and make sure that our model is accurate and generalizable in different areas.

What we did in our project is a simplified prediction. We didn't include all accessible trip data from DIVVY Chicago website due to the hugh amount of rows. Instead, we select trips from a random week as our training set. To test the generalization of our model. We select a weekday and a weekend day in the following week as test set.

2. Data preparation - Tranformation of Trips

Our initial data is the original records of every divvy trips, which include the start station/time, bike id, arrive station/time and trip duration. How can these separate trips become useful variables? Before doing the data transfromation, we need to thoroughly understand what our model is expected to achieve, and what information is needed.

As we have mentioned above, we want help the users to get rid of special cases such as bike shortage in a the near future, or to indicate possible chances of finding bikes in those "net-departuring" stations. In this case, we should use hourly summarized information of DIVVY trips, especially by their departure time.

We first read the trip data from August 1st to 10th, which incluedes both training set and test set.

```
trip <- read.csv("0801_0810.csv") #the original data downloaded from website and select time range by star time
names(trip)
```

```
##  [1] "trip_id"           "starttime"         "stoptime"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"
```

We want to know how is the starttime recorded, so we check it by looking at the fist row.

```
trip$starttime[1]
```

```
## [1] 8/10/2015 23:59
## 12865 Levels: 8/1/2015 0:00 8/1/2015 0:01 8/1/2015 0:02 ... 8/9/2015 9:59
```

The information means we can use the "**starttime**" and **from_station_name** to do the summary. The start time is recorded in the form of "mm/dd/yyyy_ho:mi", and we can use *mdy_hm* function of *lubridate* package to identify the format of time and extract month, day and hour from it.

```
start <- mdy_hm(trip$starttime)
mo <- month(start)
da <- day(start)
ho <- hour(start)
b <- as.data.frame(start)
final <- cbind(trip, as.data.frame(mo), as.data.frame(da), as.data.frame(ho))
final[1,c(2,13:15)]
```

```
##        starttime mo da ho
## 1 8/10/2015 23:59  8 10 23
```

Then we use *group_by* function from *dplyr* package to summarize hourly departure by different stations.

```
departure <- final %>%
  group_by(from_station_id, from_station_name, mo, da, ho) %>%
  summarize(
    departure = n()
  )   #summarize the departure
departure[1:10,]
```

```
## # A tibble: 10 x 6
## # Groups:   from_station_id, from_station_name, mo, da [1]
##    from_station_id  from_station_name    mo    da    ho departure
##              <int>             <fctr> <dbl> <int> <int>     <int>
##  1               2 Buckingham Fountain     8     1     7         2
##  2               2 Buckingham Fountain     8     1     8         6
##  3               2 Buckingham Fountain     8     1     9         9
##  4               2 Buckingham Fountain     8     1    10         6
##  5               2 Buckingham Fountain     8     1    11         6
##  6               2 Buckingham Fountain     8     1    12        19
##  7               2 Buckingham Fountain     8     1    13        22
##  8               2 Buckingham Fountain     8     1    14        10
##  9               2 Buckingham Fountain     8     1    15         2
## 10               2 Buckingham Fountain     8     1    16         3
```

```
dep_total <- final %>%
  group_by(from_station_id, from_station_name) %>%
  summarize(
    departure = n()
  )  #summarize the total depature by station
colnames(dep_total)=c("from_station_id","from_station_name","departure")
```

We can see that the hourly summarized departures are not continuous. In certain hours, such as 12pm to 6 am, there isn't any departures, but the *group_by* function wouldn't notice them as "0" departure because this function only counts the number of rows.

So we have to create a new dataframe that include these zeros. We create a continuous list of NA departures and use *merge* function to replace some of the zeros from our gouping summary.

```
station <- levels(trip$from_station_name) #find all the station mentioned in the trips
station1<-data.frame( from_station_name = station)
station2 <- merge(dep_total, y, by.x=c("from_station_name"), by.y=("from_station_name"), all.y=TRUE)
station2 <- station2[,-c(3)] #get a list of stations and their ids

x<-data.frame(hour=seq(0:23)-1) #create hour list
z <-data.frame(day=seq(1:10)) #create day list- August 1st to 10th
timeline <- merge(x,f1,all=TRUE) #create a continuous timeline
w <- data.frame(departure=NA)
NALIST <- merge(w,f2,all=TRUE) #create a continuous timeline with NA departure

f3 <- merge(NALIST, station2, all=TRUE) #merge the station list and the NALIST, and create a "NA" list to

colnames(f3)=c("departure","ho","da","from_station_name","from_station_id")

outcome <- merge(departure ,f3 ,by.x=c("ho","da","from_station_name","from_station_id"),by.y=c("ho","da","from_station_name","from_stati
on_id"),all.y =TRUE)  #replace the NA in column departure by the summarized non-zero departures

departure2 <- data.frame(outcome$from_station_id,outcome$from_station_name,outcome$da,outcome$ho,outcome$departure.x) #writing a new lis
t to sort the data

colnames(departure2)=c("station_id","station_name","day","hour","departure")

departure2[is.na(departure2)] <- 0 #replace the NA departure with zeros because they are equally important observations.

departure3 <- departure2[order(departure2$station_id ,departure2$day, departure2$hour ),] #sorting the data
```

Using the codes above, we create a list of continuous departures from August 1st to 10th by hour and station, which is exactly the dependent variable we would explore in our project.

3. Model selection and variables of time lag.

Our dependent variable is departures for each selected bike station in every hour from August 1st to 7th in 2015, which is24192 samples in total. The independent variables consist of three categories: spatial variables, social and economic characteristics for each census tract, and time variables. Data sources come from opendata Chicago and ACS 5 year estimate. In our projects, we use OLS method to build our predictive model.

Part1: Spatial variables:

1.**Distance to bike lane**

2.**Distance to bus stations**

3.**Length of bike lane in each census tract**

4.**Bike lane density in each census tract**

5.**Average distance to the nearest three landmarks**

6.**Distance to parks**

7.**Distance to historical preservation sites**

8.**Rack density (in the buffer of each station)**

9.**Distance to metro stations**

10.**Street density in each census tract**

Part2: Social and economic characteristics for census tracts:

1.**Median household income**

2.**Number of households**

3.**Percent of bachelors' degree in tract population**

4.**Total population**

5.**Percent of population commute by car**

6.**Mean time to work**

7.**Percent of population below poverty line**

8.**Male population**

9.**Median age**

10.**Median household value**

11.**Median gross rent for tracts**

Part3:Time variables:

1.**The total departure for each hour in Chicago**

2.**Daily weather: including temperature, humidity, rain intensity, total rain, wind speed, barometric pressure, solar radiation**

3.**Hourly departures for the selected bike stations in the last week**

4.**Hourly arrivals for the selected bike stations in the last week**

5.**Time lag departures 1: Departures one hour ago**

6.**Time lag departures 2: Departures two hours ago**

For the hourly departures and arrivals in the last week, we use the same approach mentioned above to get the data from July 25th to 31th.

For the last two *"Time lag departures"*, we use the *lag* function of *dplyr* package.

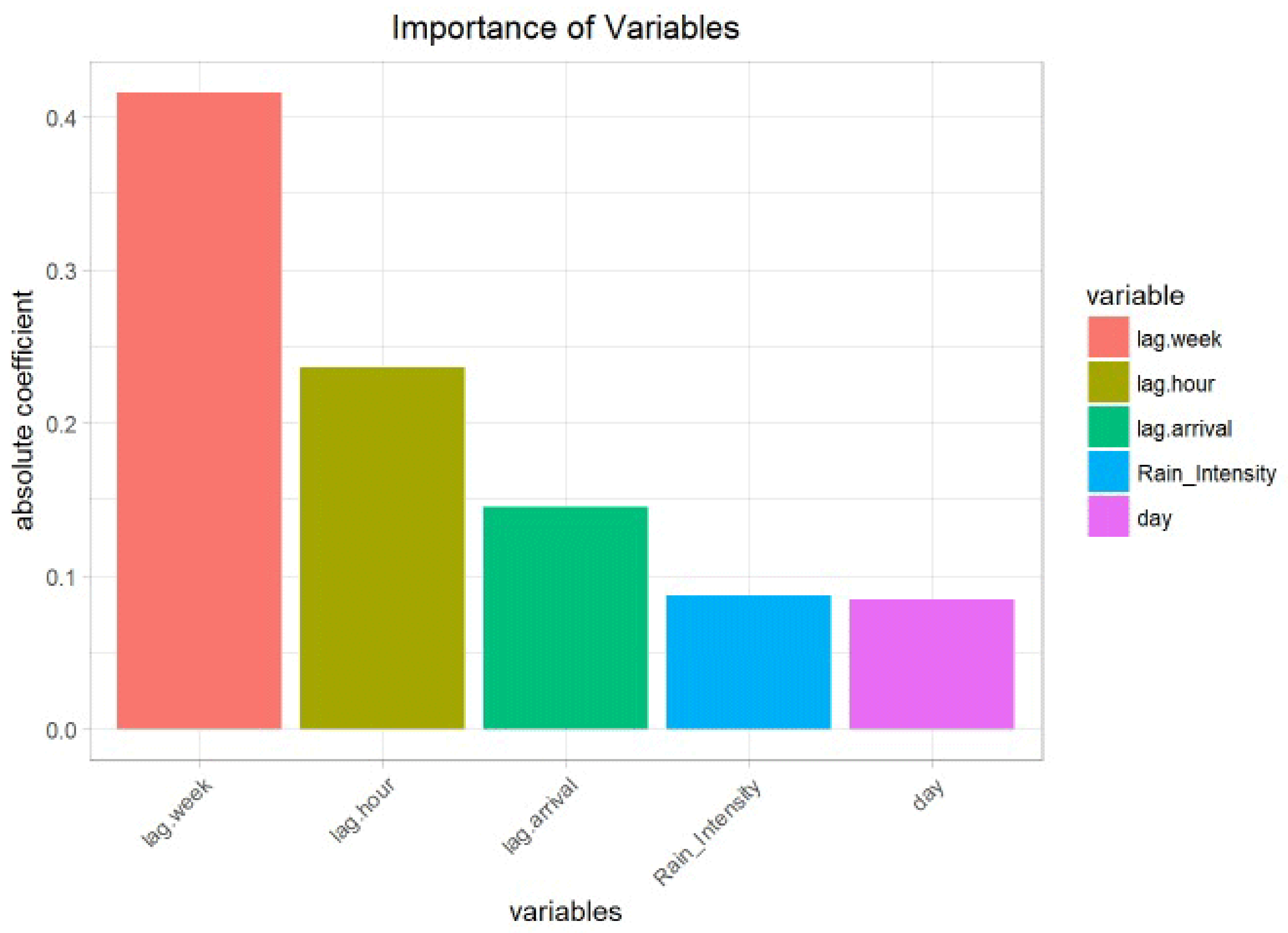
```
depature3 <- departure3 %>%
  group_by(station_id) %>%
  mutate(lag.hour = dplyr::lag(departure, n = 1, default = NA)) %>%
  mutate(lag.hour2 = dplyr::lag(departure, n = 2, default = NA))
```

4. Prediction model

We build and compare two different models for our prediction. In the first model, we use all the variables mentioned above, while in the second model, we exclude all the spatial and social-economic characteristics of census tracts, and replace them with the station names of each bike station. Then we run the regressions and the 10-fold cross validation. By comparing the two models, we discover that although the r square of the first model is a little bit lower, the mean absolute error is also lower as well. Thus, in this case, we decide that the first model is better for the prediction, since we want our model to be more generalizable.

R square	MAE	Regression
0.6910018	1.721398	regression 1
0.6941898	1.738036	regression 2

The result shows that among all the variables, the five most significant variables are all time variables, including the hourly departures and arrivals of the same day in the last week, the departure an hour ago, rain intensity of that day, and the number of day.



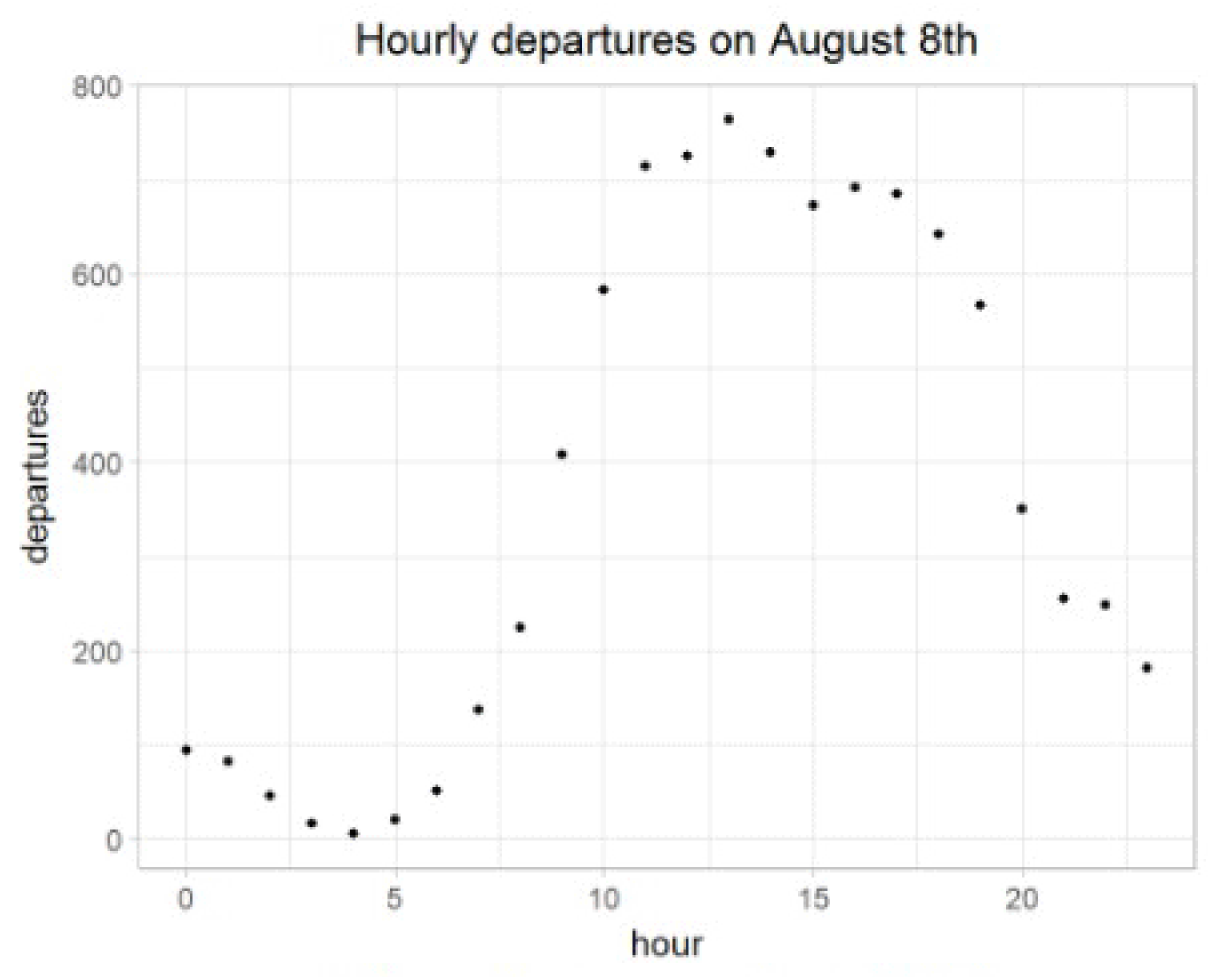
5. Model validation

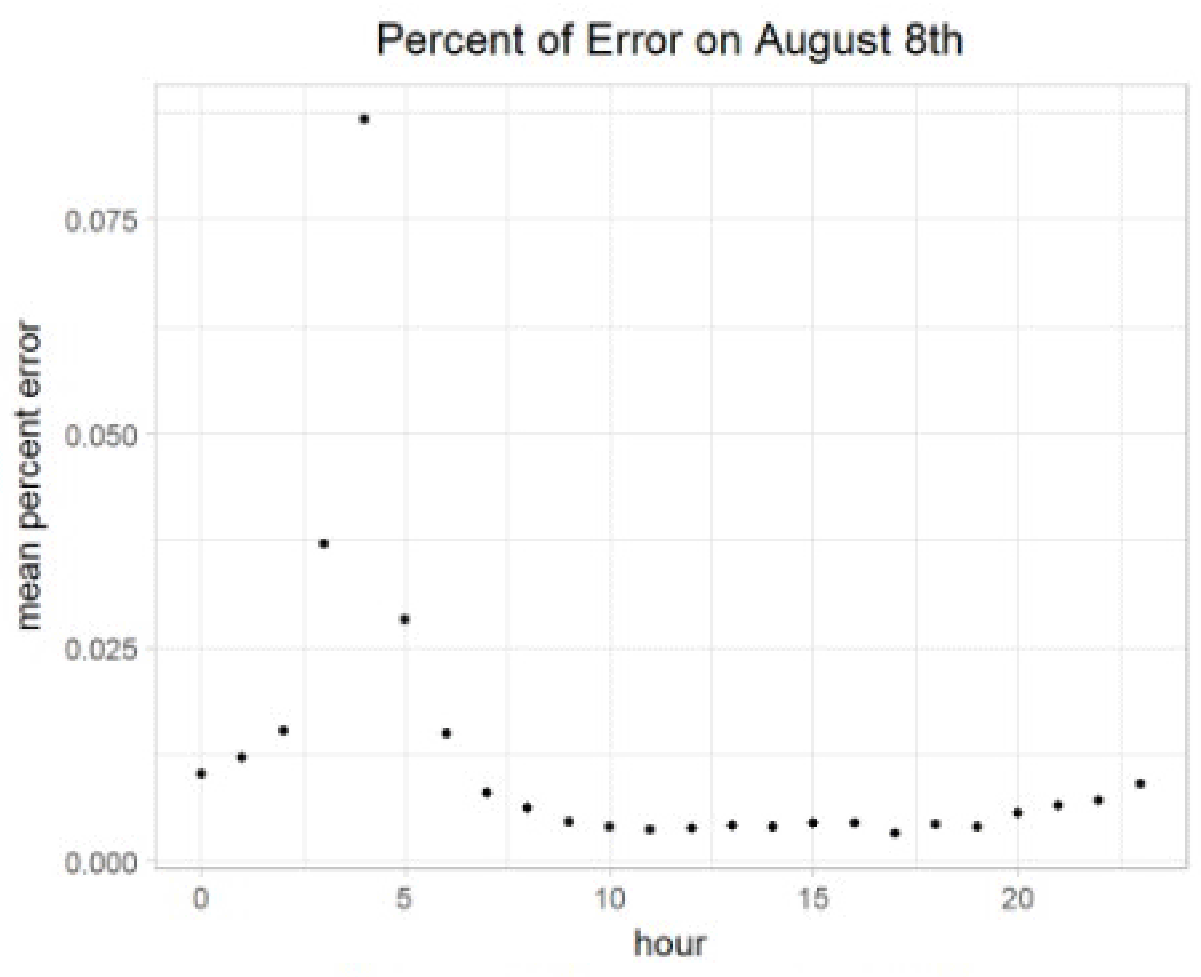
To test if our model is generalizable for both the weekday and weekend, we use this model to predict for one weekend, Aug.8th, and one weekday, Aug 10th.

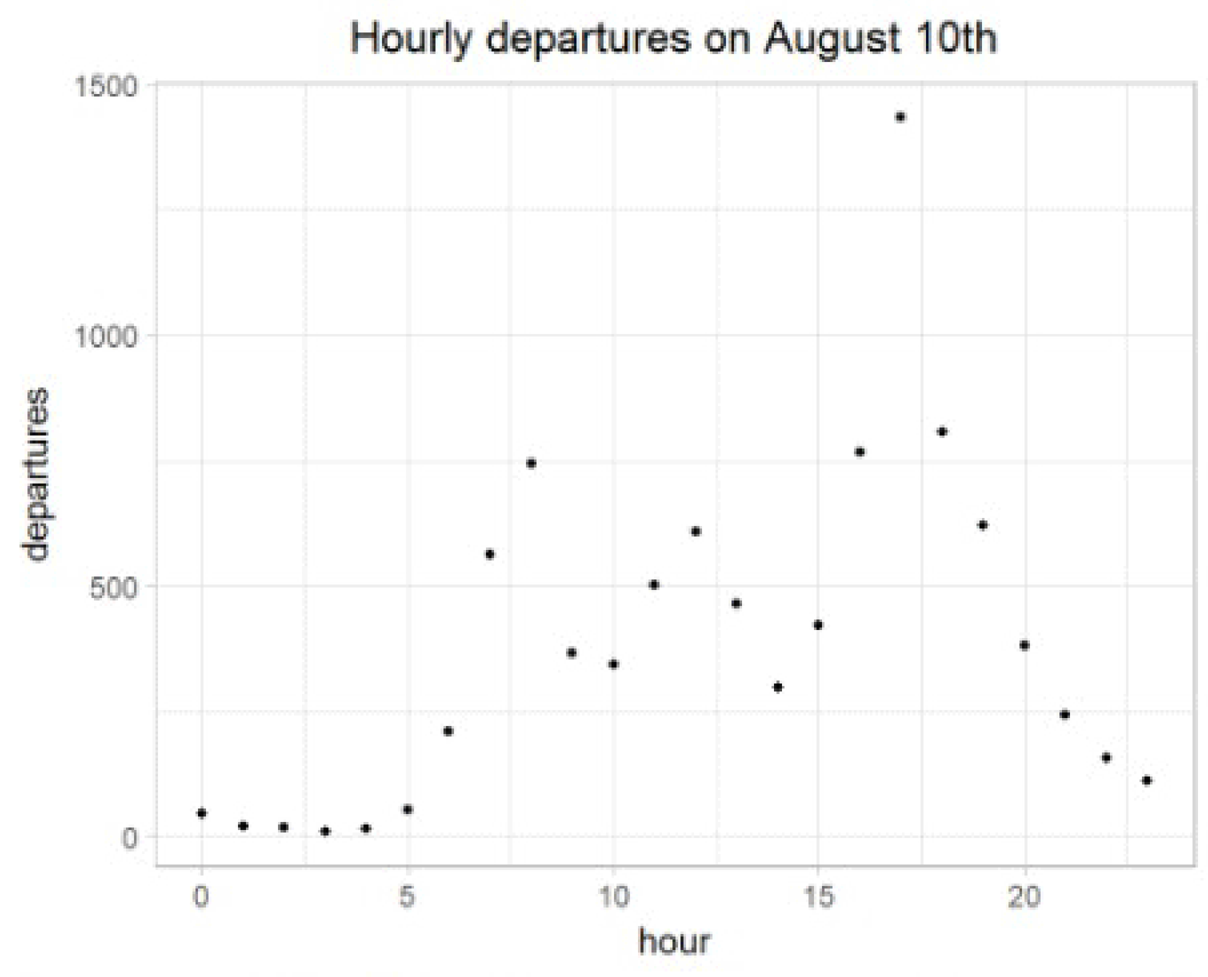
For the training set, some of the time variables are based on real-time information, for example, the weather of that day. However, we are not able to know the weather when we are predicting for someday in the future. So for the test set, we assume that these variables are the same as the same day of last week: (1) Weather of the predicting day (2) The hourly departure of the whole city in the predicting day (3) The time lag departures: departures one hour and two hours ago

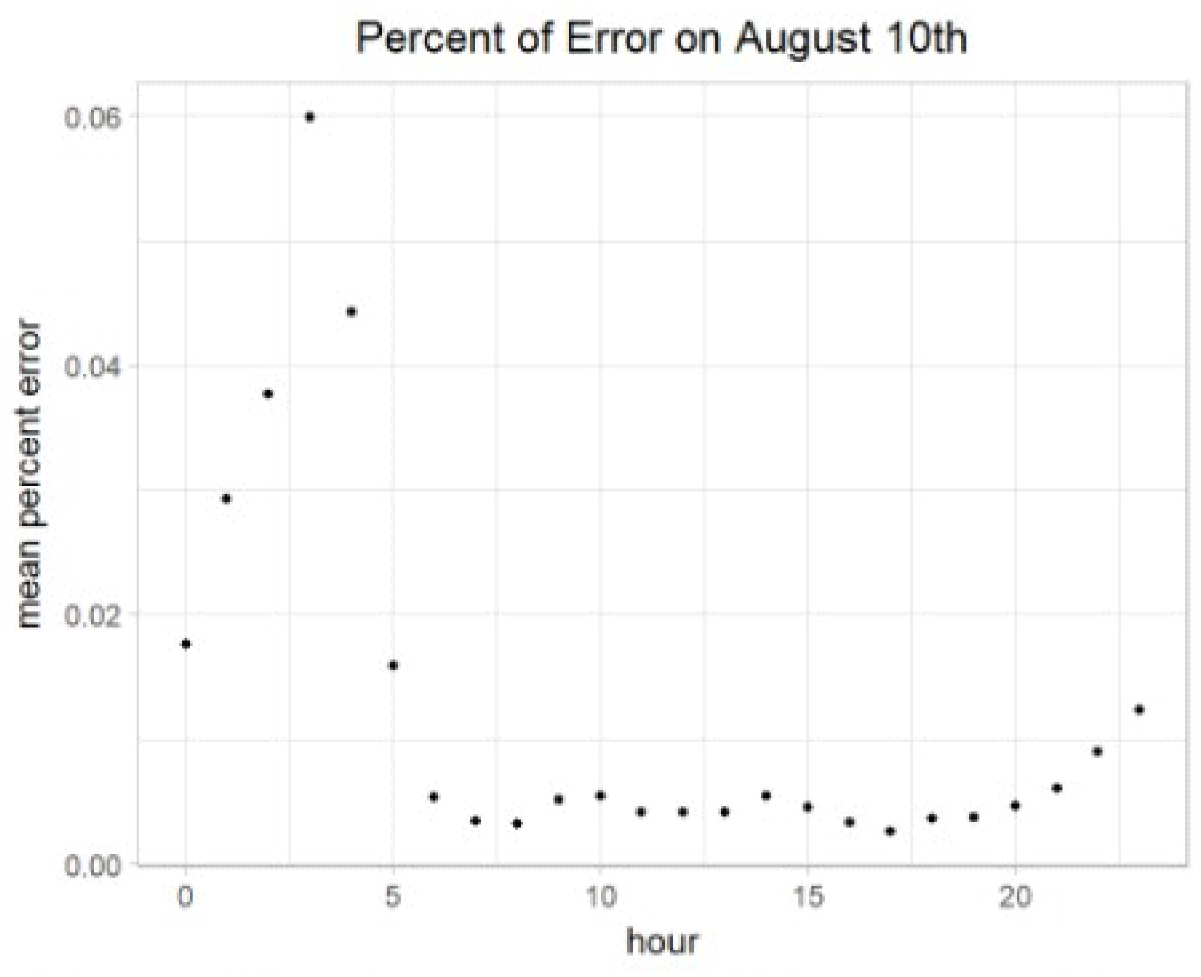
For the predicting days, we calculate the mean absolute percent error (MAPE) for each hour to test if our model is doing well in different time of one day: MAPE = mean absolute error of all stations/total departure of all stations

The result shows that the percent of error is larger at night for both the weekday and the weekend, but it is much lower and more stable during the day time, which means that our model is doing well at hours with more departures. This is the outcome we want, since typically, more people use bikes in daytime, and fewer people use bikes at midnight.



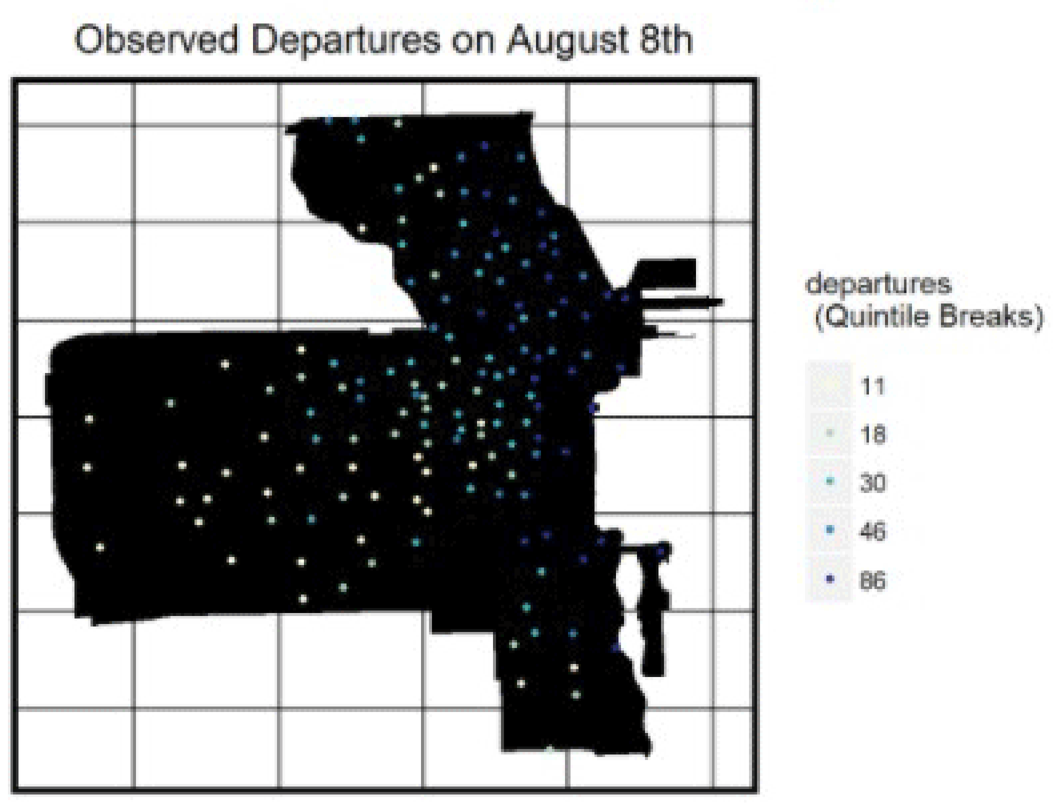


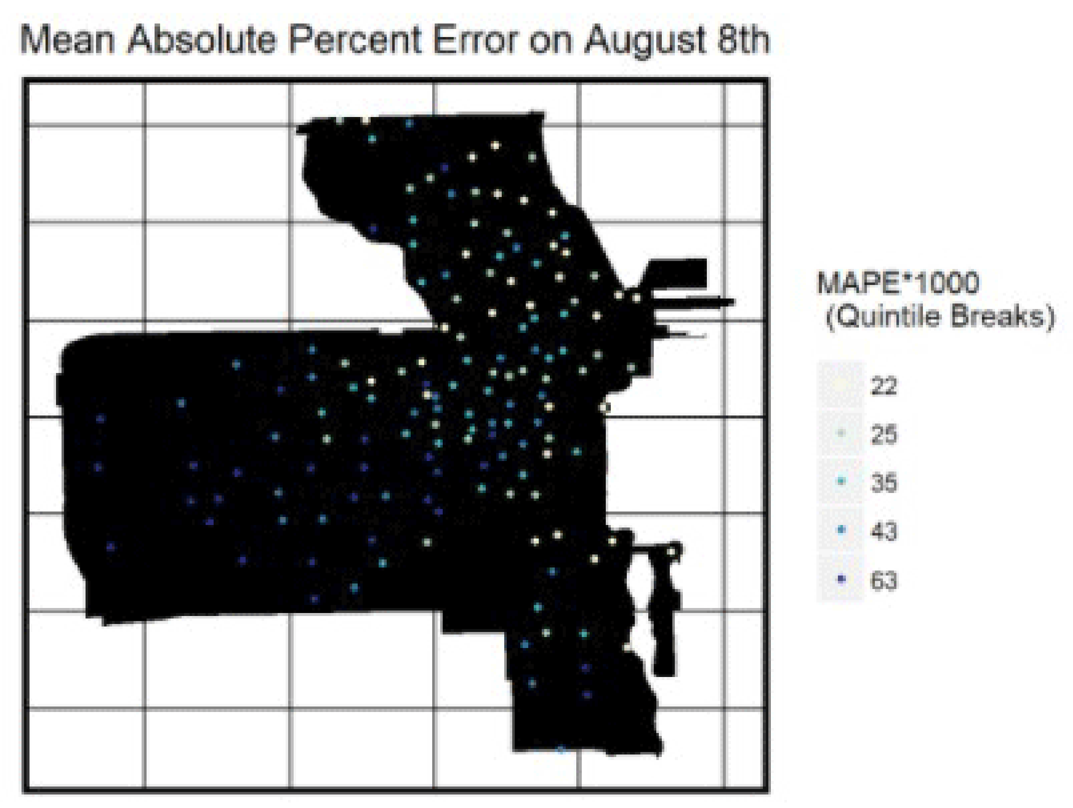


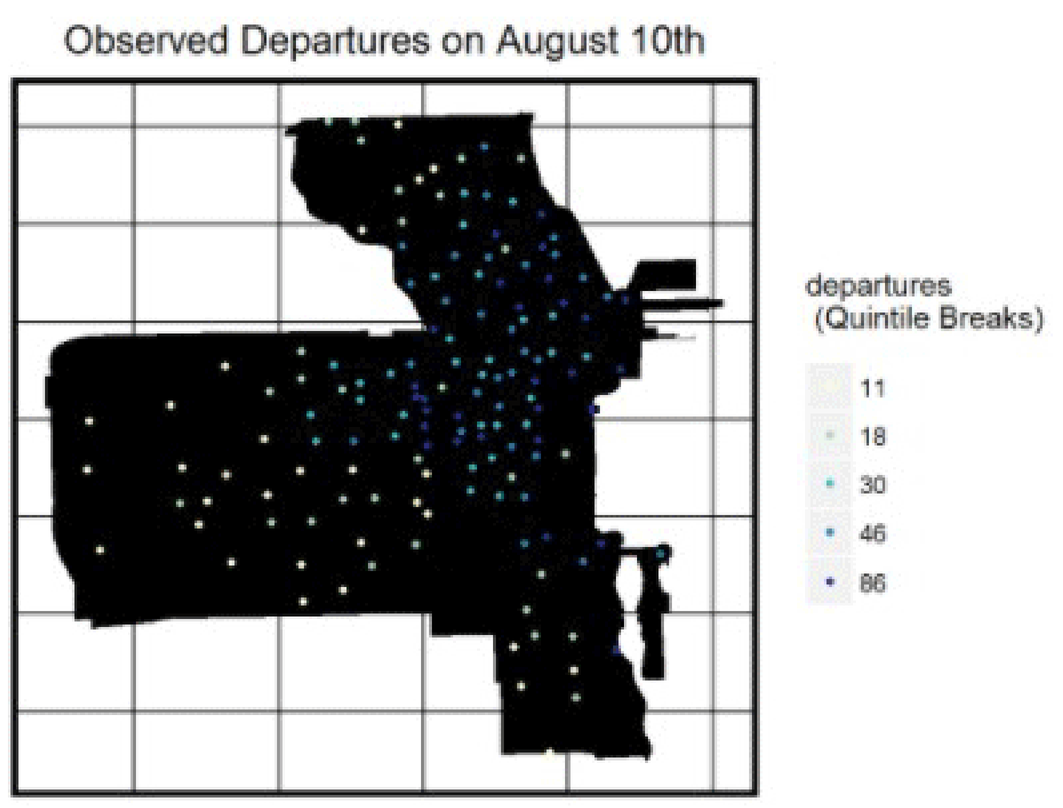


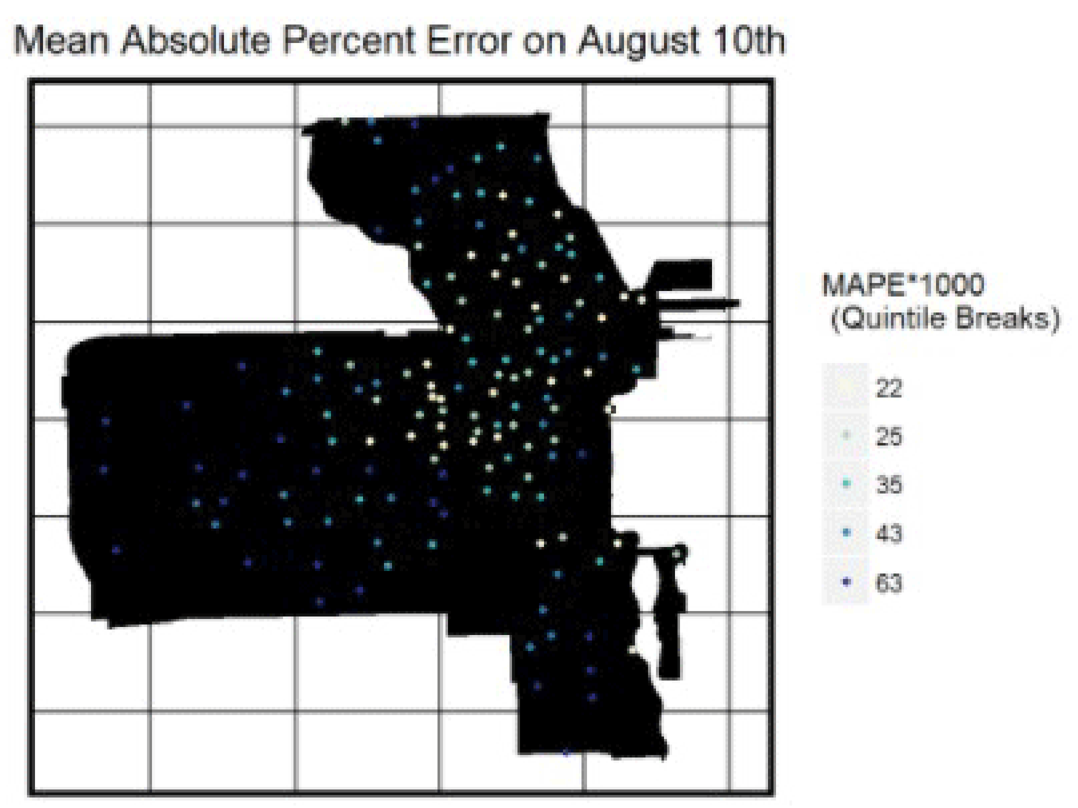
We also map the MAPE for each station to see if our model is doing well for all stations: MAPE = mean absolute error of all hours in one day/total departure of all hours in one day

The result shows that the percent of error is larger for stations with less departures, and lower for stations with more departures. This also means that our model is doing better at stations with more departures.







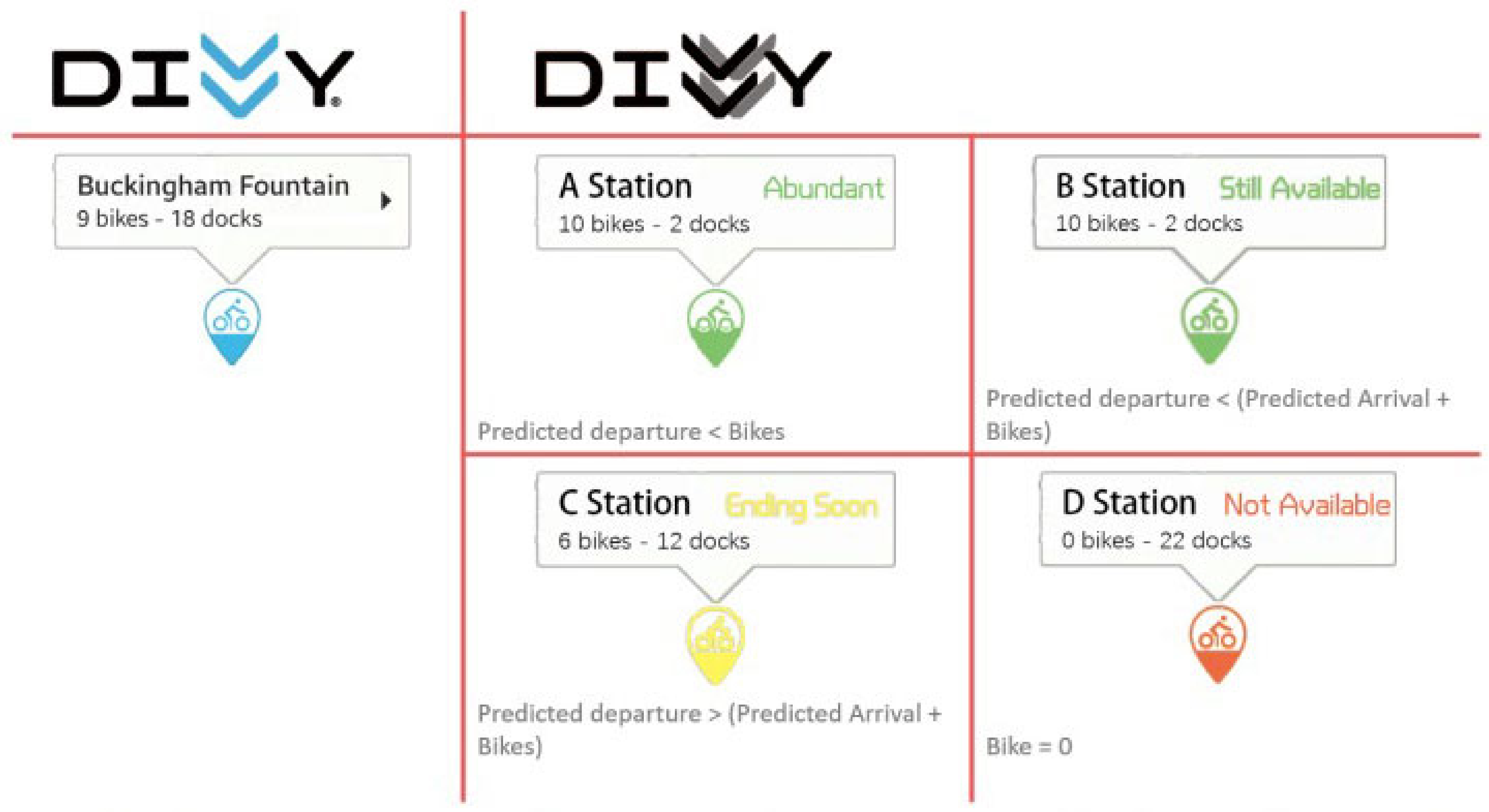


In conclusion, our model is valid and useful in the real world, since it is comparatively doing well at predicting hours and stations with more departures, and normally more frequently used stations during busy time are more in need of bike rebalancing.

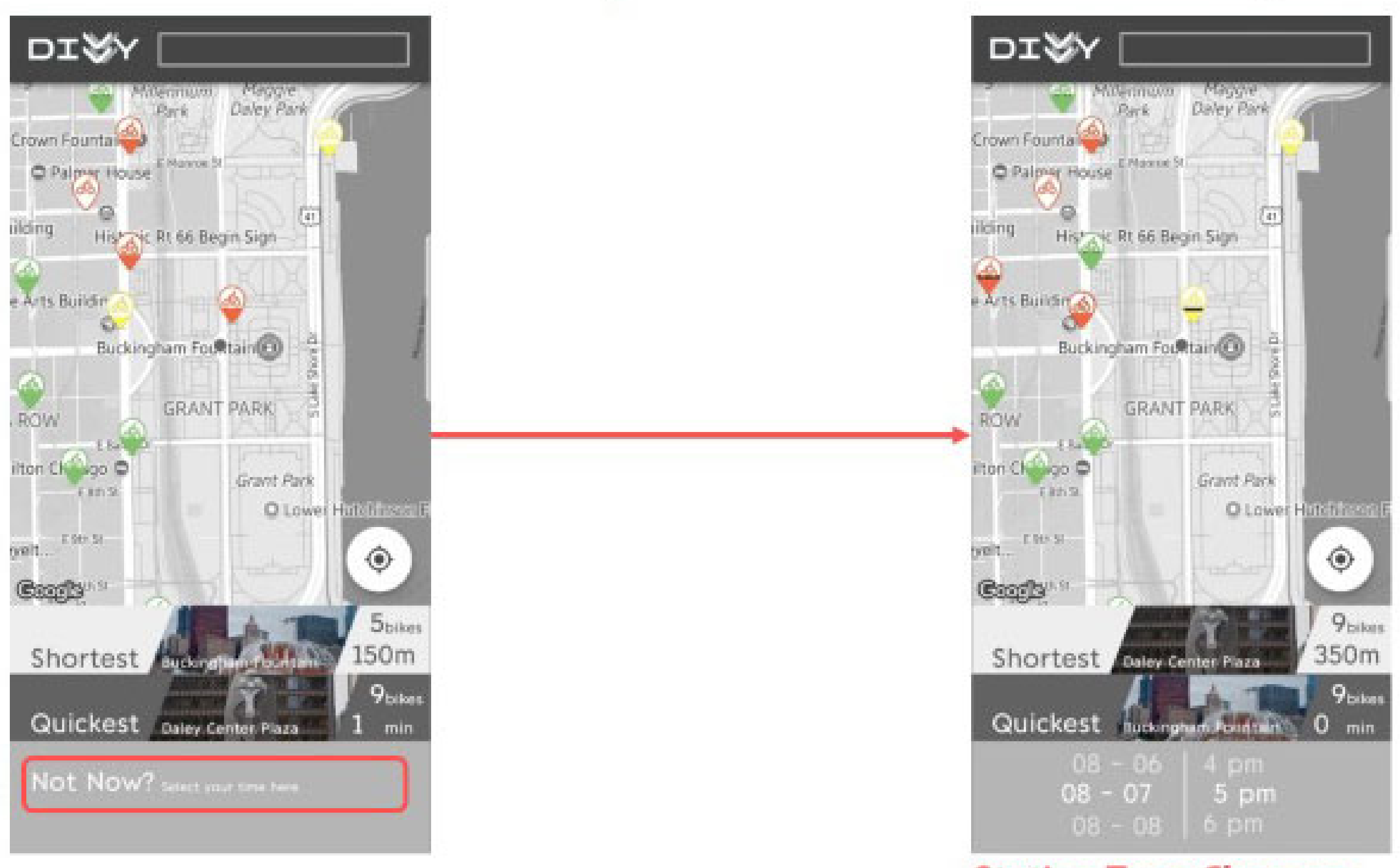
6. Usage of the model

In this case, we are predicting the departures for each hour. This model can also be used to predict the bike arrivals. By combining the prediction of departures and arrivals, we will know how many bikes and docks available in the next few hours, and therefore can provide more information to the bike users.

Imagine there is an app which can tell you how many bikes available in different stations within the next few hours, how the available bikes will change in the future, how long should I wait to get a bike, and what is the most time efficient trip. This new app we are designing is called Divvy plus, which will create smarter, more time efficient divvy trips for all the bike users. When you open our Divvy plus, you will see that the bike stations are categorized as three different colors. The red station means that there is currently no bike available, the yellow station means that there are bikes available now, but there will be no bike in the following hour, and the green station means that there is and will be bikes available in the following hour.

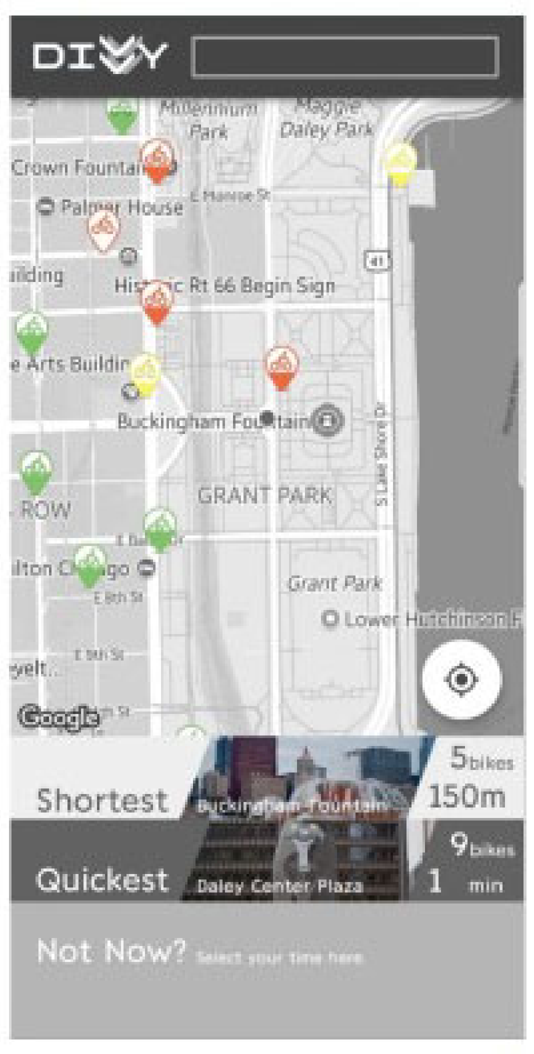


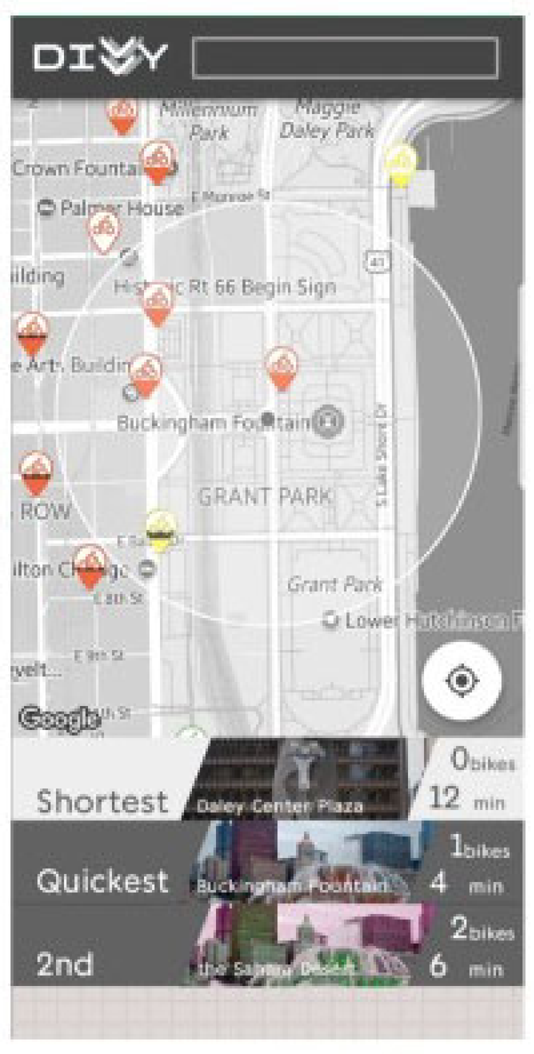
This app is also able to balance the usage of bikes in the future. If users want to use a bike, the app will search for the nearest available stations. If the nearest station is currently no bike available, it will calculate your wait time for the next available bike. It will also offer an alternative choice by searching for the nearest bike station with bikes currently available and calculate the cost of time to get there.



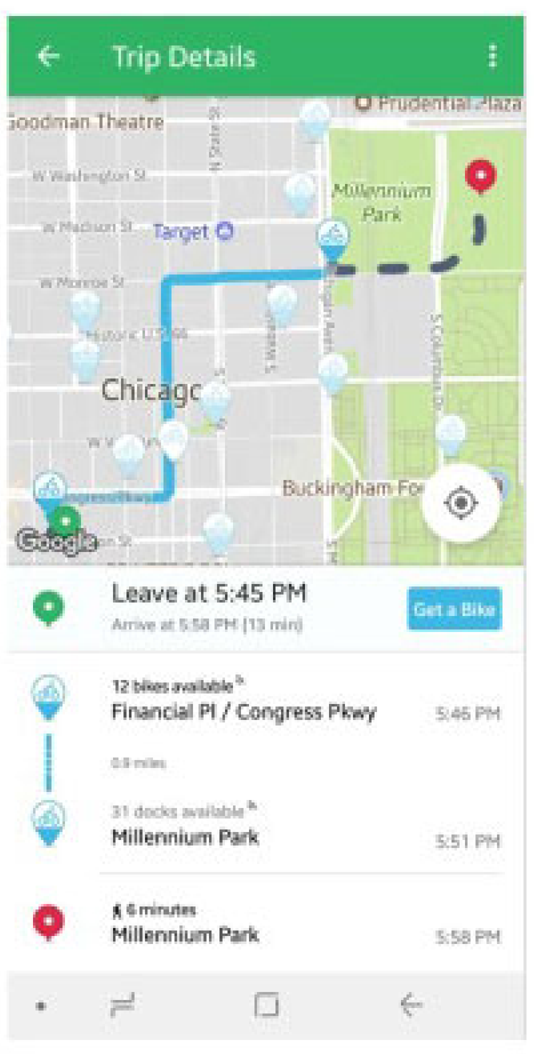
Station Type Changes

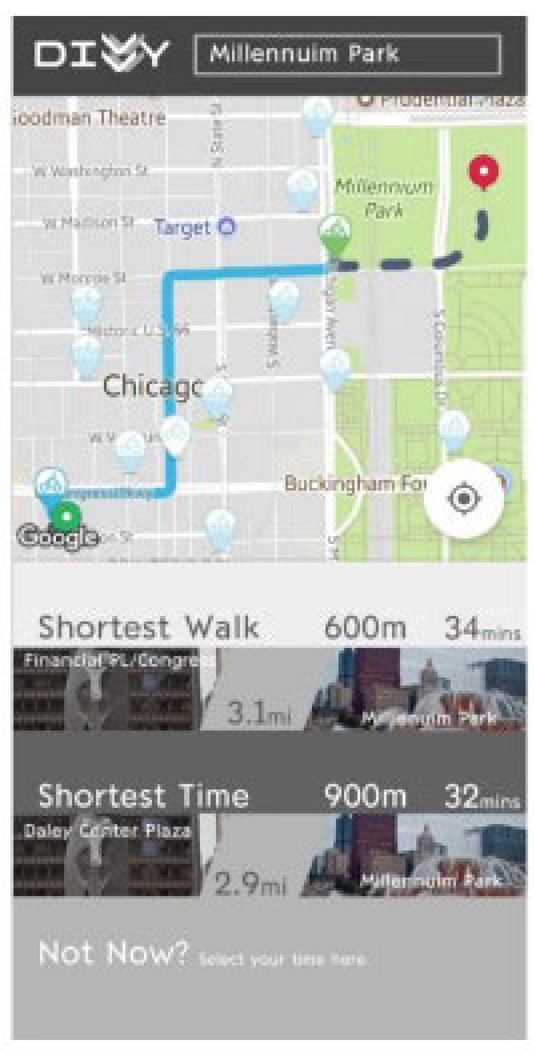
What if all the bike stations around you will not be available in the next hour? Our Divvy plus will then calculate the average waiting time for you. It will also provide a list of all the wait time of your nearby stations, and calculate the distances and cost of time to get to the station, so that you can choose your trip between the shortest walk and the shortest cost of time.





What if you arrive at your destination, and find that there is no available dock for you to park your bike? When typing in your destination in our app, it will search for the nearest bike station from your destination with docks available, so that you will know where you can park your bike at the end of the trip. It will also calculate the total cost of time of your trip and provide different trip options, so that you are able to plan for your trip in advance.





7. Improvement

Our model and app still need improvement. We can include more samples and more historical data in our model. If we include departures from the last few weeks and months, our model will be more accurate and generalizable. We can also let the users to report error on our app when we fail to predict departure for some stations, and include that error report as a variable in our model. In this way we can improve the accuracy of our model by engaging our users.

By Keep riding