

# To Wait or Not to Wait?

Predicting the arrival-departure frequency of bikes at CitiBike stations

May 12, 2020



**Data Science for Business Analytics, Technical**

**Group 5**

Sonal Sharma, Andrii Malai, Eileen Hengel, Wanting Xi

## Table of Contents

|                               |           |
|-------------------------------|-----------|
| <b>Business Understanding</b> | <b>3</b>  |
| <b>Data Understanding</b>     | <b>5</b>  |
| <b>Data Preparation</b>       | <b>11</b> |
| <b>Modeling</b>               | <b>13</b> |
| <b>Evaluation</b>             | <b>15</b> |
| Logistic Regression           | 15        |
| Decision Tree Classifier      | 16        |
| K-Nearest Neighbors           | 17        |
| <b>Deployment</b>             | <b>20</b> |
| <b>Appendix A</b>             | <b>24</b> |
| <b>Appendix B</b>             | <b>26</b> |

# Business Understanding

## **Project Description:**

The supervised model will predict the frequency of CitiBike riders taking and docking bikes at a particular station at any point in the day. We plan to create different models for a select number of stations and different models for inflow of bikes (how many bikes will arrive) and outflow (how many bikes will leave). The different models will be triggered based on whether a dock is full (outflow prediction) or empty (inflow prediction), as soon as the customer doesn't need these predictions in other cases.

## **Business Problem:**

CitiBike stations can be a ten minute or more walk apart. If a rider arrives at an empty or full station, he/she is faced with the decision, do I wait for a bike to arrive; do I wait for a bike to leave; or do I take another mode of transportation? There are costs associated when riders have to wait for bikes or wait for docks. It is likely that inconveniences created by these scenarios keep some potential riders from using CitiBike, especially for riders who frequently use popular stations. Additionally, in the case when no bikes exist at a station, many riders may give up and choose to walk or take another mode of transportation. The prediction model will enhance customer experience and help CitiBike capture unrealized demand by giving riders who would normally opt out the opportunity and information to determine whether to wait for a bike.

What if we could help riders make this decision by showing them the estimated time to the moment when a bike will arrive or depart from the station? This will give riders the information to make decisions that will impact whether they are late for work or an appointment. Furthermore, it would save a rider the time and effort of walking to another CitiBike station.

Finally, the model could help CitiBike predict where they will run out of bikes on any given day. CitiBike likely already has models that predict usage, but managing the

allocation of thousands of bikes across the city is a time-consuming and costly endeavor. If CitiBike is filling bikes at stations that don't actually need it, then the bikes could be directed to other stations. By predicting the frequency of a rider arriving, CitiBike could more accurately determine whether stations need to be manually replenished or can be organically replenished.

**Business Solution:**

By predicting the frequency that bikes arrive and leave a station the CitiBike app would display the estimated time when a bike would arrive at or leave the station in a given time period.

The target variable for the inflow model (that predicts the time when the next bike will be docked at the station) will be the category based on the number of bikes arriving at the dock station at the specific hour of the day. We introduced three categories for the target variable:

- 0-5 bikes per hour,
- 6-12 bikes per hour,
- more than 12 bikes per hour.

The prediction of the category would be converted into the estimated time needed for the next bike to arrive at the dock station.

To predict the target variable, we have a number of attributes/features including calendar date, hour of the day, temperature and precipitation level. The same structure was created for the outflow model as well. By modeling the target class, we hope not only to save riders' time but also to capture unrealized demand in the case where potential riders opt not to use CitiBike because there are no bikes at the current station.

## Data Understanding

The core of the data for the model is sourced directly from CitiBike. CitiBike makes trip data available to the public via their website<sup>1</sup>. The data spans from 2013 to 2020 and a new file is posted every month.

There are 15 columns in the Citi Bike data set which breaks down into the following fields:

- Trip Duration (seconds)
- Start Time and Date
- Stop Time and Date
- Start Station Name
- End Station Name
- Station ID
- Station Lat/Long
- Bike ID
- User Type (Customer = 24-hour pass or 3-day pass user; Subscriber = Annual Member)
- Gender (Zero = Unknown; 1 = Male; 2 = Female)
- Year of Birth

Once the core data set was identified, we worked to analyze the data and understand whether all of the features were necessary for our model.

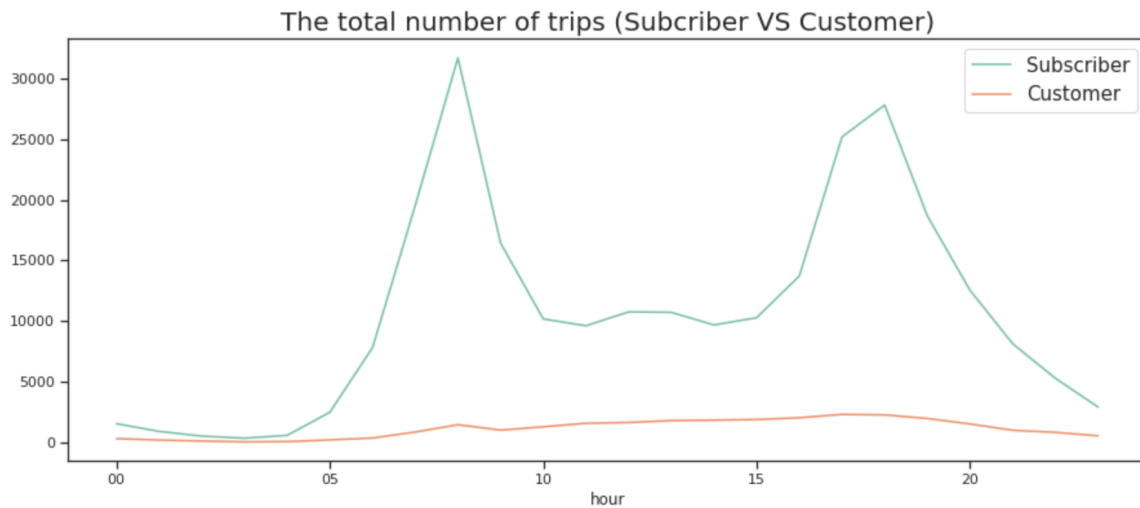
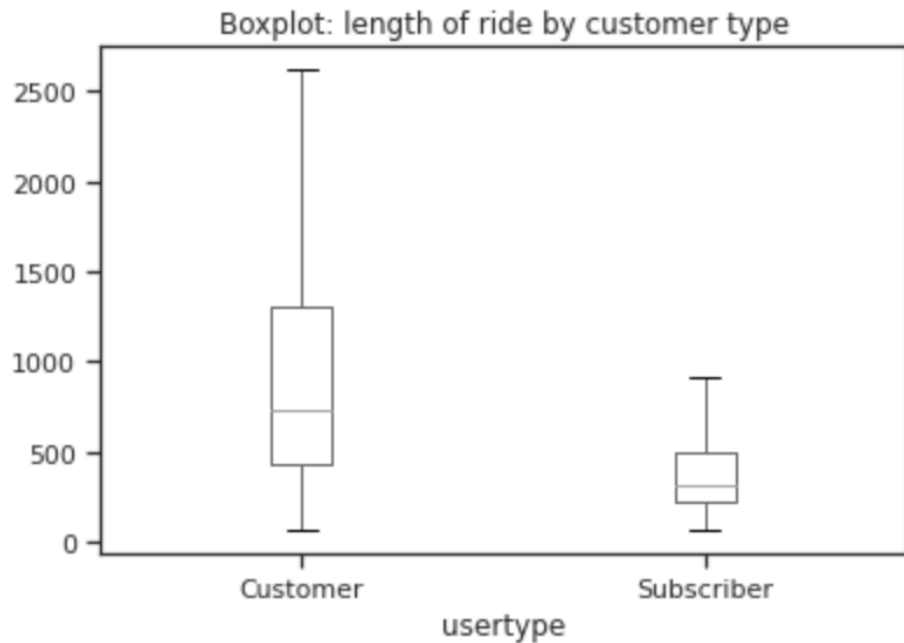
User Type:

- In order to understand how customers were using CitiBike, we looked at the length of trip by usertype. A customer is an individual who does not have an annual pass (i.e. subscriber).

---

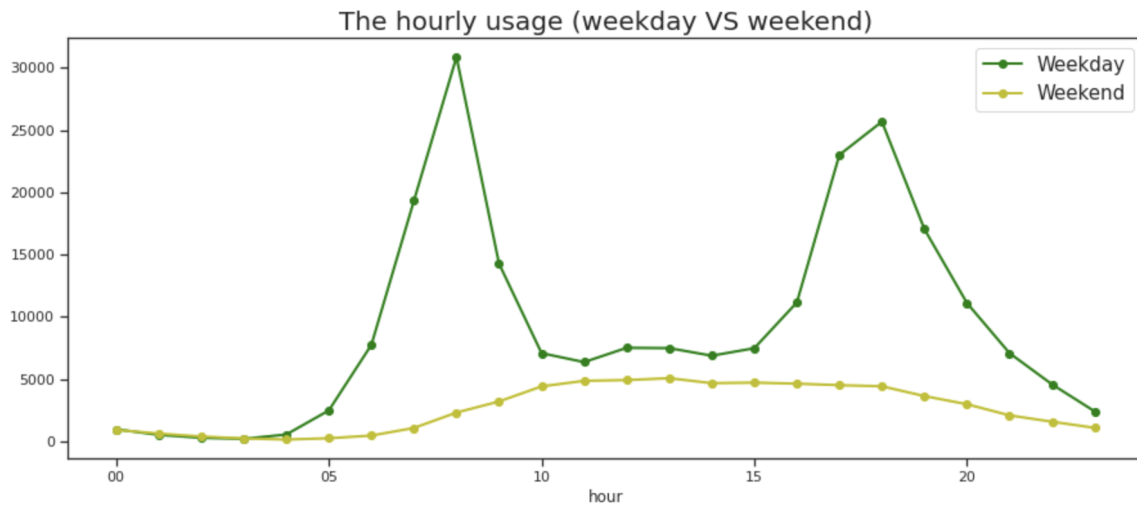
<sup>1</sup> <https://www.citibikenyc.com/system-data>

- Subscribers use the bike for shorter periods and as later analysis shows, at higher frequency. This will help us understand who will be the target customer for the model.



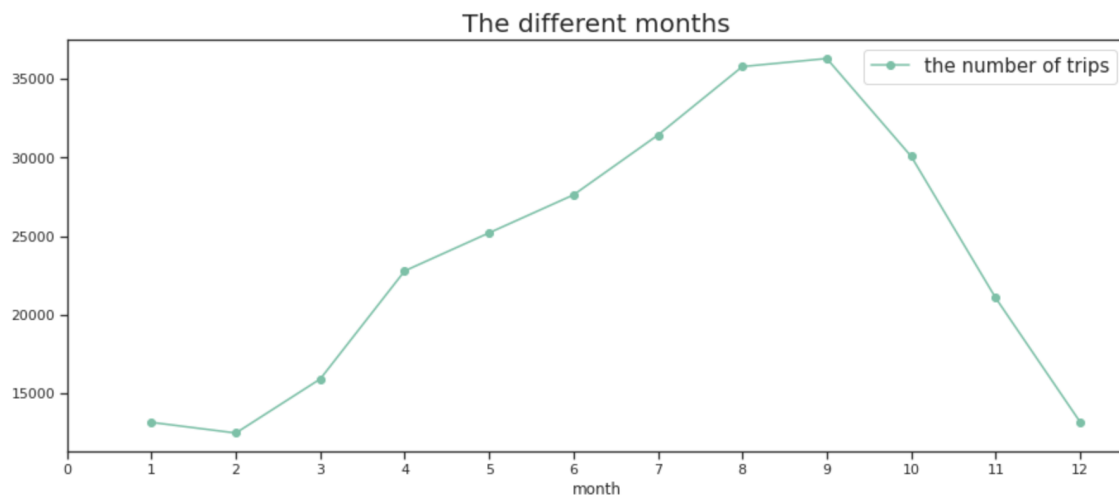
- There are two peaks of the trips of the subscriber but no peak for the customers.
- We can conclude that most subscribers use Citi Bike for commuting, and as for customers, it is used for personal reasons.

## Weekend and Weekday Trip:

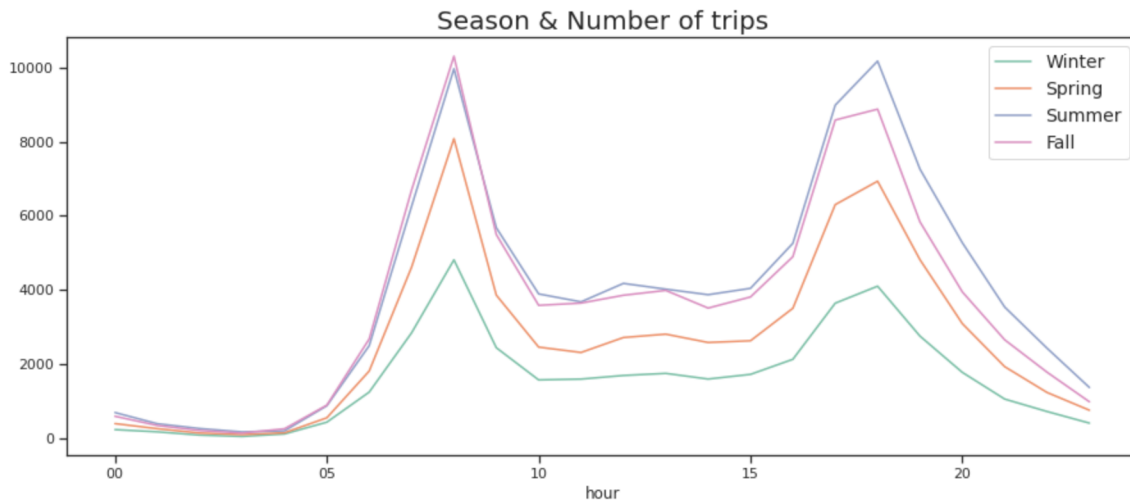


- On Weekdays, the bicycle usage has two peaks: 8:00 to 9:00 and 17:00 to 18:00. This correlates to commute times. There is also a small increase at noon.
- On weekends, there are no dramatic peaks but there is a slow crescendo to 13:00.
- From this analysis we derived that the majority of trips are used for commuting.

## Monthly and Seasonal Analysis



- There are more trips in warmer months
- The rise could also indicate an overall increase in subscribers.



- There is no variation by season for the pattern of usage, only the amount.

Based on our analysis, different customer types didn't show different use patterns. Subscribers outpaced casual riders (customers), so it wasn't necessary to differentiate the two. Also, gender, DOB, and bike ID had no correlation to the frequency of rides at a particular station. For this reason, we dropped those fields. Our analysis did show, however, that seasonality and time of day had significant impacts on the amount of riders using CitiBike. Therefore, we decided the CitiBike data would need to be augmented with additional information in order to accurately predict the target variable. Furthermore, from the data set, we spliced a number of fields based on our understanding of the business problem. Specifically, we separated the dates by hour and month to better analyze the data.

New feature columns created:

- Date (by splitting 'starttime')
- Hour (by splitting 'starttime')
- Month (by splitting 'starttime')
- Day (Value from 0 to 6 : Mon to Sun)



- Isweekend (Value 0: Weekday; 1: Weekend)
- Isholiday (Value 0: Not a holiday; 1: Holiday)
- Isworkingday (Combine Isweekend and Isholiday, Value 0 and 1)

Secondly, in order to analyze the data, we created four fields. To start, we designated any day that was a weekend and any day that was a holiday. Ultimately, the combination of these two columns signified whether a day was a working day. Using our understanding of the business problem, we anticipated that workdays (non holidays) would impact ridership and thus impact the frequency that bikes arrive or depart a station.

Based on previous research performed by Mark Martinez there is a direct correlation between CitiBike ridership and the amount of precipitation on a single day<sup>2</sup>. According to Martinez, “.. individuals are deterred from riding bikes under more precipitation, extra snowfall, and higher wind speeds” (12). For every degree increase in temperature, ridership will increase by approximately 500 riders and for every inch increase in precipitation, ridership decreases by over 8,000 riders (Martinez, 5). In the case of wind speed, every mph increase windspeed decreased ridership by 320 rides. Based on Martinez’s analysis, temperate and precipitation had the highest effect on ridership with precipitation having the greatest impact on ridership numbers.

Based on this information, we determined that weather and precipitation were key features required to make our model successful. For precipitation and temperature, data was sourced from NOAA<sup>3</sup>.

CitiBikes are located in four of the five New York City boroughs (Manhattan, Queens, Brooklyn and the Bronx) and weather can fluctuate across the boroughs, but not drastically. Secondly, most trips end or start in Manhattan as riders are going to work or leaving work. For this reason, we used the average temperature in Central Park on a


---

<sup>2</sup> Martinez, M. (2017). The Impact Weather Has on NYC CitiBike Share Company Activity. *Journal of Environmental and Resource Economics at Colby*. Retrieved from <https://digitalcommons.colby.edu/>

<sup>3</sup> <https://www.ncdc.noaa.gov/cdo-web/datasets>

given day. To come up with the average temperature for the date we used the following formula: average temp = (TEMP MAX + TEMP MIN)/2.

In the end, we selected data from the Central Park weather station only for 2019. The resulting dataset that was later appended to the main dataframe as columns COLD and PREC, as seen below:



|     | NAME                        | DATE       | PREC | COLD | IsWorkingday |
|-----|-----------------------------|------------|------|------|--------------|
| 0   | NY CITY CENTRAL PARK, NY US | 2019-01-01 | 0.06 | 48.5 | 0            |
| 1   | NY CITY CENTRAL PARK, NY US | 2019-01-02 | 0.00 | 37.5 | 1            |
| 2   | NY CITY CENTRAL PARK, NY US | 2019-01-03 | 0.00 | 40.5 | 1            |
| 3   | NY CITY CENTRAL PARK, NY US | 2019-01-04 | 0.00 | 41.0 | 1            |
| 4   | NY CITY CENTRAL PARK, NY US | 2019-01-05 | 0.50 | 44.0 | 0            |
| ... | ...                         | ...        | ...  | ...  | ...          |
| 360 | NY CITY CENTRAL PARK, NY US | 2019-12-27 | 0.00 | 50.0 | 1            |
| 361 | NY CITY CENTRAL PARK, NY US | 2019-12-28 | 0.00 | 47.0 | 0            |
| 362 | NY CITY CENTRAL PARK, NY US | 2019-12-29 | 0.25 | 41.5 | 0            |
| 363 | NY CITY CENTRAL PARK, NY US | 2019-12-30 | 0.74 | 39.0 | 1            |
| 364 | NY CITY CENTRAL PARK, NY US | 2019-12-31 | 0.02 | 40.5 | 1            |

365 rows x 5 columns

There is a risk associated with using a daily temperature average instead of the hourly temperature actuals, however. Temperatures can vary greatly across a single day; what might be considered comfortable in the morning, would become unbearable in the afternoon. However, we didn't have access to hourly temperature history. Moreover, we believe that seasonality can also be used to offset the discrepancy in ridership in the morning versus the afternoon.

## Data Preparation

Our current data frame contained all of the rides taken in 2019 at every station along with the average temperature on the given day and the precipitation on that day. To cleanse the data set, first we looked for null values. CitiBike did a good job of cleaning their own data prior to posting it so there were no null values; however, there were real-life scenarios that we wanted to exclude from our data because we felt it would skew our results. For example, very short rides indicate that the bike was faulty. If there's only one broken bike at a station, in essence there's no bike. Therefore, we chose to exclude rides that were less than 90 seconds and had the same start and end station.

As noted previously, there is high correlation between the day of the week and the number of rides. More specifically, whether it's a workday or not. To determine whether it was a workday or not, we identified instances that are both a workday and not a US holiday in order to designate days like Memorial Day.

Additionally, the CitiBike data set is very large. Creating a model that encompassed all stations seemed risky and unrealistic. Therefore, we chose to build a model off of a single (top) station.

### Top Station Analysis:

- The CitiBike data stores both start station and end station and the top station (by number of transactions) is different for each variable. We believe this is because people only take one-way trips (i.e. only ride home from work). The stations seem to operate with the hub and spoke model wherein riders take bikes from highly concentrated work centers (e.g. Grand Central, Fidi) and ride them home to residential neighborhoods.
- The top CitiBike Station is Pershing Square North (#519). For our first model, we will only look at this station. Pershing Station North is located at Grand Central Terminal.

Next, we created the column 'outflow'. The outflow summed the number of bikes that leave a station in a given hour. Each hour of the day was assigned the specific class according to its outflow: 0-5 bikes per hour, 6-12 bikes per hour, and more than 12 bikes per hour. The dataset was used to create a model for predicting the outflow of bikes from Pershing Square North Station based on the following features.

df\_model

|      | date       | hour | outflow | COLD | PREC | IsWorkingday | month | grouped_outflow |
|------|------------|------|---------|------|------|--------------|-------|-----------------|
| 0    | 2019-01-01 | 0    | 2.0     | 48.5 | 0.06 | 0            | 1     | 0-5             |
| 1    | 2019-01-01 | 1    | 0.0     | 48.5 | 0.06 | 0            | 1     | 0-5             |
| 2    | 2019-01-01 | 2    | 0.0     | 48.5 | 0.06 | 0            | 1     | 0-5             |
| 3    | 2019-01-01 | 3    | 1.0     | 48.5 | 0.06 | 0            | 1     | 0-5             |
| 4    | 2019-01-01 | 4    | 0.0     | 48.5 | 0.06 | 0            | 1     | 0-5             |
| ...  | ...        | ...  | ...     | ...  | ...  | ...          | ...   | ...             |
| 8755 | 2019-12-31 | 19   | 4.0     | 40.5 | 0.02 | 1            | 12    | 0-5             |
| 8756 | 2019-12-31 | 20   | 1.0     | 40.5 | 0.02 | 1            | 12    | 0-5             |
| 8757 | 2019-12-31 | 21   | 0.0     | 40.5 | 0.02 | 1            | 12    | 0-5             |
| 8758 | 2019-12-31 | 22   | 1.0     | 40.5 | 0.02 | 1            | 12    | 0-5             |
| 8759 | 2019-12-31 | 23   | 3.0     | 40.5 | 0.02 | 1            | 12    | 0-5             |

8760 rows x 8 columns

*Note: the data frame includes the same number of instances than hours in a year. All instances with NaN values for the labels were assigned to a 0-5 class of outflow.*

A second model was created to predict the inflow of bikes for Pershing Square North Station. The inflow model uses the same features.

Finally, let us explain the logic behind creating 3 classes of inflow/outflow. We decided to consolidate the data into three groups that will make our predictions functional for a customer. Based on our personal experience and talking to other

customers, we found that the longest a rider would wait for a bike is 10 minutes; therefore, we divided our predictions into the following categories:

- 0-5 (bikes per hour): It's likely the rider would have to wait longer than 10 minutes
- 6-12 (bikes per hour): A rider would have to wait up to 10 minutes
- More than 12 (bikes per hour): A rider would have to wait at most 5 minutes

Within our test data set, the distribution of outflow by category was not skewed.

```
outflow
0-5          3571
6-12         1586
More than 12  3603
Name: grouped_outflow, dtype: int64
```

The inflow of the test data contained similar patterns:

```
grouped_inflow
0-5          3469
6-12         1655
More than 12  3636
Name: grouped_inflow, dtype: int64
```

## Modeling

Our target variable, the class of the instance according to the inflow and outflow at stations, showed strong correlation to weather and seasonality. In order to choose a model that captured the correlation and because we had a classification problem, we focused on logistic regression, decision tree classifier and k-nearest neighbors. Each model offers unique benefits and drawbacks. One common benefit of all these models

is that it's easy to train/test them and tune the complexity level so that we will not result in overfitting.

Logistic Regression computes the probability that an instance belongs to each class using the trained logistic regression classifiers which picks the class with the highest probability and returns the class label. As part of feature engineering, we have trained our models with only those features which were relevant towards prediction of our target variable. A negative of the model is that it cannot solve nonlinear problems since its decision boundary is linear and it is vulnerable to overfitting due to sampling bias.

The decision tree classifier has posed benefits due to the low level of data preparation needed (e.g. no normalization) or scaling. Decision tree has the ability to distinguish classes without large errors. As a result, there are fewer misclassification errors. It also has the benefit of being quite easy to explain to stakeholders. The drawbacks of the model are that it likely won't be able to generalize all stations as small changes in data will have large impacts on the classification model.

Finally, the KNN model provided benefits in that data could be added at will without a need to re-train the model. It performs on spot and instance based learning. However, the disadvantage of KNN is that it is computationally costly on large data sets and is expensive in terms of real-time execution.

The models will solve the business problem by predicting the classes of the instance (a specific hour of the day) according to the inflow and outflow at stations. As we mentioned before, this will help to send 1 out of 3 kinds of functional messages through the Citi Bike app:

- You will need to wait longer than 10 minutes
- You will need to wait up to 10 minutes
- You will need to wait up to 5 minutes

This will equip riders with information so that they can decide to wait for a bike to arrive or leave, walk/ride to another station or take a different mode of transportation. If the model correctly predicts that another bike will arrive or leave in five minutes, riders

will save the time and effort required to bike or walk to another station. Typically riders choose stations closest to their destination. If they have to walk/ride to a station 10 minutes away, this could add 20 minutes to their commute (10 minutes to walk to another station, 10+ minutes to walk back to their destination).

The classes are predicted based on the following features of instances:

- Month
- Hour of the day
- The average temperature during the day
- The precipitation level on a given day
- Whether the day is a working day or not

To review Colab files, refer to [Appendix B](#).

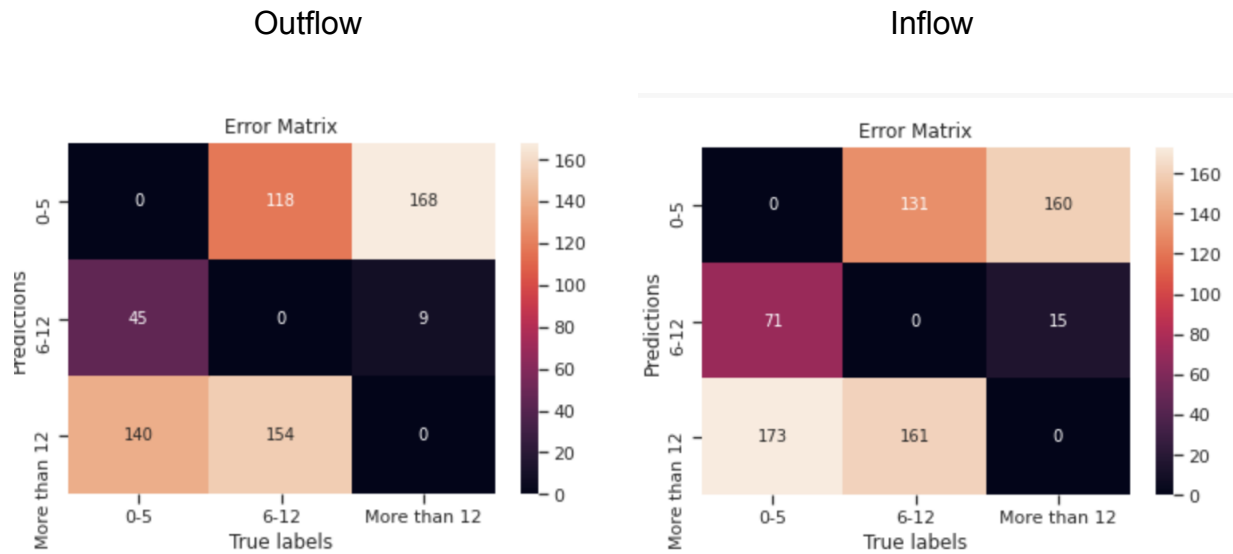
## Evaluation

To evaluate the performance of models, we looked at confusion matrices and then multiplied them by the cost matrix that was based on our business knowledge.

### Logistic Regression

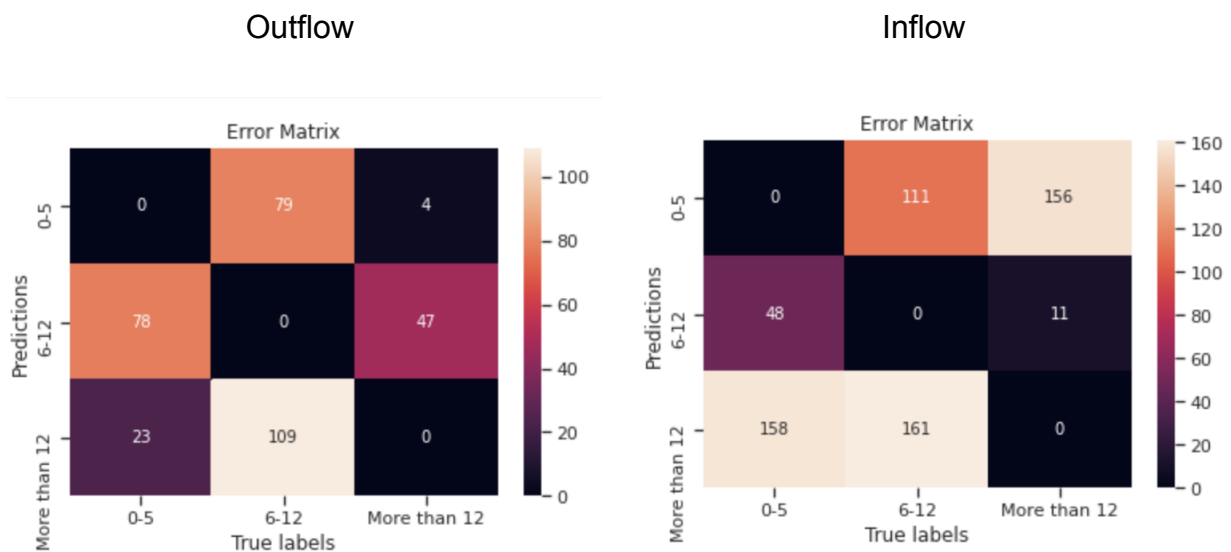
After breaking the data into test and train sets, we first trained the data in a logistic regression model which returned a 63% accuracy. The confusion matrix exposed a number of errors. The model was consistently mis-classifying instances when only 0-5 bikes came in an hour as more than 12 in an hour. Additionally, it was also mis-classifying when more than 12 bikes came as 0-5 bikes.

Based on the costs associated with mis-classification - especially when the model predicts that a rider will wait five minutes or less (More than 12 bikes) and they end up waiting more than 10 - we determined logistic regression was too costly. We saw the same trends with the inflow model. Both models were determined to be worse in terms of the accuracy compared to other types of models.



## Decision Tree Classifier

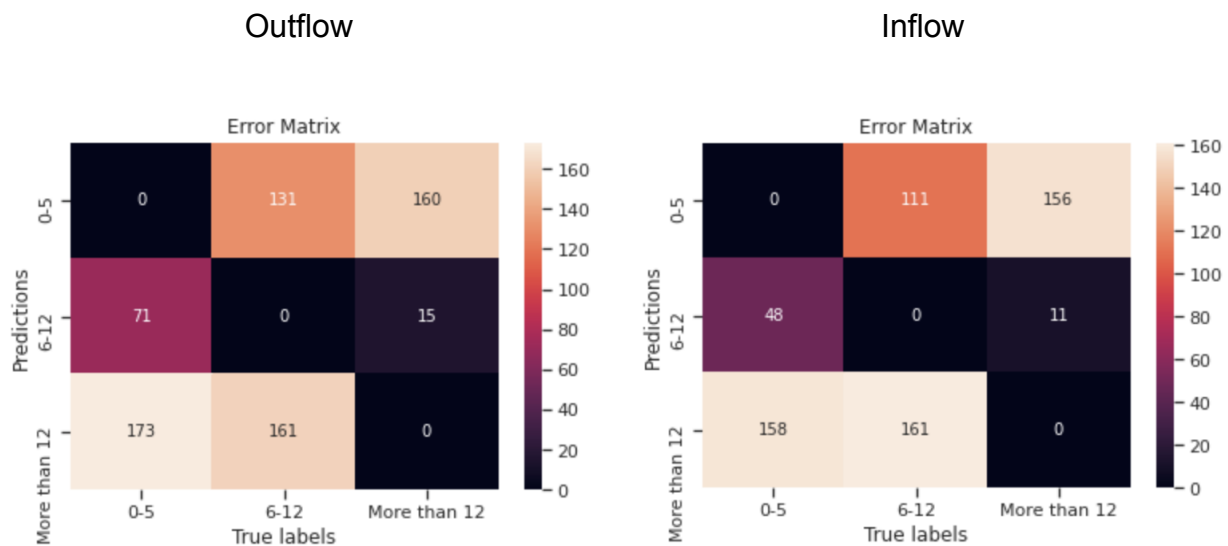
Next, we trained the data against a decision tree classifier model with a max depth of nine (as determined by the balance between the accuracy and the overfitting). Below the confusion matrix for the inflow model. Compared to the logistic regression, the number of errors is significantly lower. The outflow model performs equally as well.





## K-Nearest Neighbors

The final model we tested was a K-Nearest neighbors. Below the confusion matrix for the inflow model. Compared to the decision tree classifier, the number of errors is higher. The outflow confusion matrix also shows an equally higher level of errors compared to the decision tree classifier.



The final decision should be made based on the expected costs of deploying one of the above-mentioned models. Thus, we have created a cost matrix based on the business logic and the assumption that the average hourly wage of a NYC resident is \$27. There are no specific financial costs related to erroneous predictions made by the model.

So how do we assign the dollar value to some errors in our cost matrix? Let's explain the logic. We want to evaluate the costs a customer will incur due to bad predictions made by our models. For instance, let's imagine the following scenario. The customer came on the bike to the dock station near his/her house. The user found that the station is full. Having opened the app, the user got the prediction that the next bike will leave the station within the next 10 minutes. The customer decided to wait for 10 minutes, but nobody undocked the bike during this time scope. Finally, the user is disappointed and decides to leave the station to dock the bike at another station. How

much time was wasted in this case? 10 minutes, which are equal to \$4.5 — based on the average hourly wage of a NYC resident. These are not direct costs but we wanted to estimate the influence of the negative user experience due to wrong predictions.

Thus, we came up with the following cost matrices for the inflow and outflow models:

|           |                    | TRUE      |            |                    |
|-----------|--------------------|-----------|------------|--------------------|
|           |                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
| PREDICTED | 0-5 bikes          | 0         | 0          | 0                  |
|           | 6-12 bikes         | 4.5       | 0          | 0                  |
|           | More than 12 bikes | 4.5       | 2.25       | 0                  |

The cost matrices are developed based on the same logic as the error matrices. The costs were computed by dividing \$27 by 60 minutes to calculate the cost per minute. The minute cost (\$0.45) was then multiplied by the anticipated wasted time (examples for outflow model are provided below):

- You need to wait more than 10 minutes for the bike to be undocked (corresponds to 0-5 bikes)
- You need to wait up to 10 minutes for the bike to be undocked (corresponds to 6-12 bikes)
- You need to wait up to 5 minutes for the bike to be undocked (corresponds to more than 12 bikes)

We can see that only three types of errors generate time waste for a user. The first scenario (column '0-5 bikes' and row '6-12 bikes') was described above. The second scenario (column '0-5 bikes' and row 'more than 12 bikes') is the same but the only difference is that the user got the prediction that the bike will leave the dock station within the next five minutes. We assume that many users will still wait extra 5 minutes, so this type of error generates the same amount of waste cost - \$4.50. The third scenario (column '6-12 bikes' and row 'more than 12 bikes') generates only half of \$4.50, because in this scenario, the rider has to wait five minutes extra to dock the bike, costing the rider \$2.25.

Why are other cells in the upper right corner zeros? First, as we mentioned before in the paper, we assume that nobody will want to wait more than 10 minutes. Thus, when the model predicts more than 10 minutes to wait the customer will go to another station or use another mode of transportation and will not waste any time. Second, in the case of the error in column 'more than 12 bikes' and row '6-12 bikes' the user gets the time gain instead of costs, because the model predicted 10 minutes wait time, and the customer waited only up to 5 minutes.

Due to the poor performance of the logistic regression, the cost matrix was not computed.

- Pershing Square North Station:

#### Decision tree cost matrix

Inflow total cost for Pershing Square North station: \$699.75

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 351       | 0          | 0                  |
| More than 12 bikes | 103.5     | 245.25     | 0                  |

Outflow total cost for Pershing Square North station: \$740.25

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 450       | 0          | 0                  |
| More than 12 bikes | 99        | 191.25     | 0                  |

#### KNN cost matrix

Inflow total cost for Pershing Square North station: \$1460.25

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 319.5     | 0          | 0                  |
| More than 12 bikes | 778.5     | 362.25     | 0                  |

Outflow total cost for Pershing Square North station: \$1289.25

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 216       | 0          | 0                  |
| More than 12 bikes | 711       | 362.25     | 0                  |

Based on the cost matrix for the Pershing Square North Station, the decision tree classifier has the lowest cost.

In order to test the model further, we chose two other stations in NYC and tested the performance of the decision tree and KNN again (see [Appendix A](#) for results). After comparing against other stations, the decision tree performed better than the KNN. We decided to choose the decision tree as our model.

## Deployment

The model was built using historical weather data (precipitation and temperature) which will not be available when the model is deployed. Therefore, the model is dependent on forecasted weather data which will create a level of uncertainty for the prediction. Once the forecasted weather data error matrix is within the margin of error of the historical weather data, the model can be deployed using forecasted data.

We plan to have the multi-phase deployment for our prediction model. First, we will test the work of our models on the limited number of dock stations in New York City. We will choose five stations that are often full and/or empty. As we mentioned before, the model will make predictions only in these scenarios, so the right choice of the stations will ensure the most rigorous testing of our model in the real environment. Moreover, it will be indicated in the app that this new feature is functioning in beta mode, and users will be able to share concerns about its functioning and post reviews on their

experience. By gathering both quantitative and qualitative data we can evaluate how well the models perform in the real-time.

If the accuracy of predictions is good enough and feedback from users is mostly positive, we will gradually increase the number of dock stations supporting this feature until it's reached all dock stations. It is very important to pay attention to the amount of data available for each station. Not all stations have the same amount of training data in a given year. As soon as we create separate prediction models for each dock station, it's important to underline that maybe the lack of the data could lead to one of the following options:

1. The scope of the data should be increased to years prior to 2019 so that the model could reach the needed level of generalization and predict inflow/outflow of bikes properly.
2. The predictions will be not available for some dock stations or the models trained on data for other similar stations will be used for such stations. For example, a new station will not have any data for proper training of the model. Thus, we could:
  - a. Turn off this function in the app until we get enough data to train/test the model and deploy it, or
  - b. Use the model for another similar dock station for a new one. We have tested this approach and found that vanilla accuracy drops by ~10% by doing so. However, stations were chosen randomly for this test.

The major risk for our project is technical one. Without testing these models in real time we can't be sure that the accuracy of predictions would be sufficient for the massive use of the prediction models. We also don't know whether this new feature will increase the NPS (net promoter score) for Citi Bikes app or increase the ROI of the business in general. That's why we need the above-mentioned gradual deployment and testing.

We also must have a “plan B”. If the accuracy of predictions is not good enough or feedback from users is mostly negative, we will need to investigate what are the reasons for poor performance and try to improve it. We have already brainstormed some ideas that could potentially enhance the performance of both models:

1. Add new useful features. For instance, we assume that the speed of wind could be one of them. To our regret, we didn’t have this data for all days in 2019 in our weather dataset, so this hypothesis was left untested.
2. Use hourly data on the temperature instead of the daily average.
3. Scale up the training data up to 2-3 years (instead of 1).
4. Use hourly data on the precipitation level instead of the daily level.
5. Scale up the model to all stations and not to create separate models for each station.

However, the above-mentioned hypothesis must be rigorously tested before the deployment and were listed just to show that there could be a place for improving the accuracy of predictions in the future.

We also want to mention in our analysis the costs related to the deployment of the prediction models developed by our team. First, the company has to handle the cost of increased computing power that will be used for making the predictions. However, decision tree models are not heavy in terms of the computing power compared to other kinds of models such as KNN, for instance. Second, to cope with technical risks CitiBike should hire at least one experienced data scientist that will be able to deploy and support the operations of the models. From the analysis of employees profiles on LinkedIn we didn’t find any data scientists in its team. Third, the company could have the increased request directed to their support staff, especially in the early stage of the deployment. This will lead to extra costs as well.

However, we strongly believe that the benefits from higher customer engagement with the app and enhanced user experience will exceed above-mentioned costs if the deployment will be done according to our multi-phase plan.

In case of the success and reaching the high levels of accuracy of predictions, CitiBike could consider licensing this technology to other docked bike-sharing or scooter-sharing companies in different geographies, as soon as it doesn't directly compete with them.

## Appendix A

- University PI & E 14 St:

### Decision tree cost matrix

Inflow total cost for University PI & E 14 St: \$864

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 337.5     | 0          | 0                  |
| More than 12 bikes | 270       | 256.5      | 0                  |

Outflow total cost for University PI & E 14 St: \$1008

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 549       | 0          | 0                  |
| More than 12 bikes | 279       | 180        | 0                  |

### KNN cost matrix

Inflow total cost for University PI & E 14 St: \$1298.25

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 333       | 0          | 0                  |
| More than 12 bikes | 774       | 191.25     | 0                  |

Outflow total cost for University PI & E 14 St: \$1239.75

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 310.5     | 0          | 0                  |
| More than 12 bikes | 702       | 227.25     | 0                  |



- Broadway & E 14 St:

#### Decision tree cost matrix

Inflow total cost for Broadway & E 14 St: \$697.50

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 342       | 0          | 0                  |
| More than 12 bikes | 99        | 256.5      | 0                  |

Outflow total cost for Broadway & E 14 St: \$652.50

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 319.5     | 0          | 0                  |
| More than 12 bikes | 99        | 234        | 0                  |

#### KNN cost matrix

Inflow total cost for Broadway & E 14 St: \$1082.25

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 220.5     | 0          | 0                  |
| More than 12 bikes | 405       | 456.75     | 0                  |

Outflow total cost for Broadway & E 14 St: \$891

|                    | 0-5 bikes | 6-12 bikes | More than 12 bikes |
|--------------------|-----------|------------|--------------------|
| 0-5 bikes          | 0         | 0          | 0                  |
| 6-12 bikes         | 117       | 0          | 0                  |
| More than 12 bikes | 355.5     | 418.5      | 0                  |

## Appendix B

[CitiBike Inflow Model](#)

[CitiBike Outflow Model](#)