NEW YORK CITY
BIKE SHARING DEMAND

DATA 1030
FINAL PROJECT REPORT

JESSE GEMAN
SIBEL KADIOĞLU
BRUCE WANG

TABLE OF CONTENTS

VISION

There were 13,845,655 recorded Citi Bike trips in New York City for the year of 2016. What information lies in that data? While the rider patterns and flows might be studied to improve the service and better understand overall traffic patterns in the city other findings might be used for entirely different purposes. Perhaps, the demographic data associated with Citi-Bike riders and the neighborhoods they travel to and from might be used to improve businesses and inform real estate purchases. The possibilities are endless.

PREDICTING SUPPLY AND DEMAND

Our initial goal was to analyze and predict bike flow patterns. Specifically, we wanted to use machine learning algorithms to predict the supply of bikes at bike stations throughout the city.

One can think of the problem as predicting the demand for Citi Bikes. When a station is full of bikes, the demand for bikes at that station is low, and the supply is high. When this is the case riders will be unable to drop bikes off at that stations. Conversely, when a station is empty of bikes, the demand for bikes at that station is high and the supply is low.

The benefits of an accurate bike supply/demand prediction model are readily apparent. Citi Bike would be better able to anticipate the needs of riders and appropriately transfer bikes from station to station in the most efficient way possible to ensure that bike demand is met. In addition, Citi Bike riders would be able to plan which stations they should go to to ensure that they can either pick up or drop off their bikes.

Our initial attempts to solve this problem were discouraging. We used regression to try and predict the net change in the number of bikes at each station over one hour intervals and were unable to get good results. However, when separately tried to predict the number of bikes coming into a station over a one hour period and the number of bike leaving a station over a one hour period the results improved dramatically. We will explain in greater detail in the methodology section.

PREDICTING DESTINATION

Predicting Citi Bike riders' trip destinations based on their demographic traits could be useful to prospective business people in search of consumer demographic information for particular neighborhoods. We used various machine learning techniques to analyze and predict where certain Citi Bike riders from specific neighborhoods were likely to travel at certain hours of the day.

The results were mixed and depended hugely on the time of day for which we were making the predictions. For example, we were able to predict which neighborhoods certain rider demographics groups would travel to in the mornings with surprisingly good accuracy (nearly 70 percent). We were less successful however, at predicting where riders were likely to go in the evenings. None of this was particularly surprising, morning riders are likely commuting to the same destination throughout the week. In the evenings people might be venturing to any number of destination for any number of reasons — their patterns are less well defined.

DATA

Our final dataset is a collection of three different data. Each component is explained in the following section.

CITI BIKE DATA

Collecting the *Citi Bike* data was trivial. It can be accessed and downloaded in CSV format on the Citi Bike website. While there is data available from 2013 through 2017, we chose to look at the data collected in 2016. This dataset is composed of 12 CSV files, each file contains one month of Citi Bike data and has roughly 1 million entries. Each of these entries corresponds to one bike 'transaction', and includes the start time, end time, start date, end date, start station name, start station id, end station name, end station id, start station longitude and latitude, the end station longitude and latitude, and the unique id for the bike used. In addition, each entry also included some rider information: user type (customer or annual subscriber), gender (male, female, or unknown), and year of birth. Since much of the data was redundant for predictive purposes (e.g. bikeid, station id, station name) it was clear we needed more data to have any chance at making accurate predictions, regardless of the problem we planned to tackle. Even more, we hoped getting non-bike-related data would allow us to find more complex patterns, than we might otherwise.

WEATHER DATA

Getting additional data was more of a challenge. Finding weather data for the days covered by our dataset seemed like an obvious place to start. We settled on *World Weather Online* and used their API to get a number (more than we thought necessary) of weather related facts for New York City for each hour of each day 2016.

CENSUS DATA

We also got four *American Community Survey (Census Bureau)* data sets (demographic data, housing data, economic data, and social data) from a third party website along with a corresponding geometric dataset with the census defined neighborhoods used in the study. This was a particular challenge, the data combined with the geometries were not easy to find. While for our particular predictions we only used median neighborhood income, we have kept the data along with the corresponding geometries and thus could make use of other ACS neighborhood findings in future models.

MERGING, CLEANING, AND DATABASING

Not only was our dataset was **big**, it was riddled with missing values, and inconsistencies, many of which we were not aware of until it was too late — meaning sometimes we had to backtrack to fix things. In addition, when it came time to combine different datasets, they were often formatted differently which required some trouble shooting. We learned the hard way and it was a challenge.

We merged and cleaned the data in several stages. All of this was done using Pandas dataframes. We added a new feature to our Citi Bike data, 'time', that gives the hour of the day during which the bike was checked out (e.g. a bike check out at 9:30pm would be given the number 2100 in this column). Doing this allowed us to merge on the hourly weather data.

We then created a *SQLite* database. The code we wrote to do this closely resembles what the code we used to create our IMDB database in Assignment 4.

At this time we were still trying to predict the hourly demand for bikes at each station. In order to do this we wrote code to calculate at the net change in bikes at each station for each hour of the day. This code was modified and incorporated into new code to allow us to calculate the number of incoming and outgoing bikes at a given station for various specified, hourly blocks of time.

Lastly, we incorporated the census data. To integrate this dataset we relied heavily on the *GeoPandas* package. Using the longitudinal and latitudinal coordinates for each station from the Citi Bike dataset we were able to write code that would determine which of the census defined neighborhoods each station was located in. We then added the corresponding neighborhood income data to our dataset, which now also contained weather data.

At this point we decided to remake our SQLite database to include the neighborhood data, and remove the weather traits we deemed unimportant. Unfortunately, due to the size of our dataset, SQLite proved too 'lite' to handle many of our queries.

Refer to the blog post about the SQL database for the nitty gritty!

METHODOLOGY

After some initial, preliminary surface level analyses, it was time to dig deeper and to start using more sophisticated techniques to glean information that can't be found by direct plotting or visualization. These more-sophisticated analyses can be categorized under the umbrella of "machine learning". We took several approaches to analyze the data, all of which shed light on a different aspect of this giant monstrosity of the bike share dataset. Clustering was performed on traffic patterns at certain stations during certain hours of the day to develop 'traffic-based' neighborhoods which provided an alternative lens of classifying stations. At the same time, we also used the census defined neighborhoods to predict which neighborhood a user would end up in, given some initial starting information about the rider and their surroundings.

To cluster the neighborhoods into their traffic defined neighborhoods, we first thought about several pieces of information about a station's *hourly* traffic which one would need for a traffic based classification: number of bikes leaving that station, number of bikes getting dropped off at the station, and the proportion of bikes entering the station to bikes leaving the station. These values were what was used to cluster the stations according to the traffic defined neighborhoods. We used *k-means* clustering because of our familiarity with the algorithm. The first pothole which we stumbled upon was the realization that the k-means algorithm clusters are based on Euclidean distance, which meant that all the features which we cluster upon had to roughly have the same notion of distance between each other. This proved to be an issue which we worked around with some ad-hoc solutions. The first problem was that the ratio of ends to starts was a ratio, and so if one had more starts, the ratio would be a decimal, and so the euclidean distance does not scale linearly. This was alleviated by taking the log of a ratio, and multiplying by a factor of 2.5 to give a 'separation' that would be seen by the k-means algorithm. Next, the number of bikes dropped off and bikes departed at a station was a huge number, which was far greater than the ratio. To alleviate this, we divided the number of starts and the number of ends at each station by the 85% percentile value for starts and ends respectively, so that we have roughly the same 'distance' between the number of starts and ends and the ratio. After this was all done, the labeled neighborhoods (through clustering) were then fed into a *Carto* map for time-of-day dependent traffic neighborhoods on a map. A CSV was also written out to describe traffic neighborhoods for each station during each interval of day.

While our clustering method created a new and interesting way of defining neighborhoods, we worked the 40 census defined neighborhood tabulation areas (NTAs) that contain Citi Bike stations to predict which neighborhoods riders would travel to.

We started by converting our non-numerical, categorical data values to integer values. We then built and trained various classifiers on seasonal blocks of the data. Across the board, our *Random Forest* classifier (with 160 estimators) did the best. Our *Decision Tree* classifier (max_depth=60, min_samples_leaf = 5, min_samples_split = 2) was significantly worse, as was our *K-Nearest Neighbor* classifier (n_neighbors = 5, leaf_size = 80, weights = 'distance'). We

did have some success with a *soft voting classifier* that gave our Random Forest classifier more weight, but it was still inferior to the Random Forest on its own and took a long time.

Regardless, all of our initial outcomes were embarrassingly bad. We can't say for sure why, but our suspicion is that there is simply too much variance in Citi Bike users riding habits throughout the day to be able to predict where they are going with any accuracy. However, when we started whittling down our dataset to a smaller subset, our results improved dramatically. For example, we predicted with 65% accuracy which neighborhood women, born after 1988, were likely to ride to between the hours of 6:00-9:00 am during the work week. (For reference, if you just guess the most common destination neighborhood given the start station the result is 37%.) We did a little bit worse when we did the same predictions for male riders. In addition, we used the feature importance functions associated with the tree based classifiers to reduce the number of traits. In the end, we predicted the destination neighborhood using 13-14 features.

Finally, we used Jupyter Notebooks widgets applications to build an interactive GUI using the results of our classifiers. It allows users to choose from a selection of 'starting' parameters (start station, time of day, hour, day of week, weather conditions, etc…) and returns a *GeoPandas* map with the neighborhood that contains the start station, and the predicted end neighborhood. It is available on our Git repo.

- A Few Takeaways

While we were quite successful in predicting where Citi Bike riders would go in the mornings, particularly, when we specified the gender and age group of the rider, our efforts to predict riders' destinations during other times of the day were a little disappointing. We thought that since we could predict where people were going for their morning commute we would also have success in predicting their destinations for their evening commutes. We were wrong.
This sort of makes sense, between 6:00 and 9:00 am people ride their bikes to work. In the evenings, when ridership is at peak levels, all bets are off. A rider might go home from work one evening and go to the bar the next. Perhaps, more people take joy rides. Who knows! Even though there is more data (more people use Citi Bikes in the evening), there is likely much more variance.

It's possible our models would have done better if we had considered more of the NTA census feature. Unfortunately, it was not feasible for our computers, which were crashing often as it was. If you, the reader, are so inclined, you can access the NTA census data on our Git/Dropbox and follow our instructions (provided in the notebook file 'ADD NTA.ipynb') to add more census data to our dataset (which we can email you…it's too big for Git).

Another problem, we approached is about finding the number of available bikes in a station. To begin with, first we defined a neighbourhood parameter. It is determined based on NYC neighbours and it is used to group stations which are in same neighbourhood. Basically, a station is assigned to its closest neighbourhood. It is calculated by using distance between neighborhoods and stations' longitude and latitude. Secondly, we defined delta parameter which

has three arguments such as hour, day and neighborhood. Delta parameter represents the difference between incoming and outgoing bikes for a given station in a given hour, day and neighborhood. Incoming bikes represent the number of bikes arriving the hour before whereas outgoing bikes represent the number of bikes leaving the hour before. Namely, higher delta means it is higher likelihood of finding a bike. We visualized this approach by using python's *Folium* library. It works as follows: if a user gives a neighborhood, hour and day, our visualization method shows each stations' delta in given neighbourhood of NYC map. Color codes (green, blue, red) in map shows different level of delta and the green one shows high delta values in contrast to red.

In the next step, instead of find delta values from the data, we aimed to predict them by using machine learning algorithms. Unfortunately, our first attempt ended with failure. Our explanation is about it that actually same delta values can represent different incoming and outgoing bikes. But by calculating only delta values cause loss of information. Therefore, our new goal was to predict incoming and outgoing bikes for a station. Our feature vectors include bike info (station id), date info (hour, day, month, full date), weather info (weather description, humidity, temperature, wind speed). Besides these features, we described a new feature called *lag*. This feature includes previous specified hour values of incoming and outgoing bike. Namely, given a station if you want to predict incoming or outgoing bikes for 4pm and if you set your lag value to 3, it means our feature vector includes incoming and outgoing bikes from 1,2 and 3 pm. Originally, we have monthly data but for machine learning approach we grouped it based on seasons. For each season, two months is used as training set and one month is used as test data. Cross validation is applied over each month of the season.

We used *linear regression*, *decision tree regression* and support vector machine regression methods. Both tuning algorithms parameters, and our lag parameter on such big data was quite time consuming. We evaluated our algorithm results by using mean absolute error and Spearman correlation coefficients. The difficulty in here was to visualize our results. Because, we make prediction for each station, for each day and for each hour which approximately means 400x365x24 rows. While keeping these results, we wanted to investigate one more prediction problem. This time, we wanted to predict sum of all outgoing or incoming bikes for all station. Namely, instead of being focused on each station's incoming and outgoing bikes, we worked on total incoming or outgoing bikes. However, we use same feature vectors, data sets and evaluation criteria for this problem. In our blog, we only showed shortened version of our results but further investigations can be found in our code.

REFERENCES

[1] http://www.citibikenyc.com/system-data

[2] https://opendata.cityofnewyork.us/

[3] https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a050.pdf

[4] https://github.com/pavelk2/Bay-Area-Bike-Share/blob/master/README.md

[5] http://toddwschneider.com/posts/taxi-vs-citi-bike-nyc/

[6] Our blog: https://1030visionaries.wordpress.com/