SI 206 Final Project Report
Kira Mintzer and Dominique Roitman
April 29, 2019

1. The goals for your project.

Our goals for this project were to access the Uber and Lyft API's and extract information regarding the prices based on specific latitudes and longitudes. Some of our initial project goals for extracting information included calculating the days and times of the week that are most expensive for each company.

After accessing our APIs and analyzing the data that we would be able to interact with, we concluded that our overarching research question/goal would be to evaluate whether the cost from a major city to a nearby suburb (within an 100 mile radius) would be greater than the anticipated cost for the same ride back (suburb to major city). To do this, we planned to gather data from 5 different major cities, and for each city, 5 nearby suburbs/smaller cities (25 pairs of longitudes and latitudes total). The goal was to successfully extract this data by calculating the average price for the rides throughout different times of day, calculating the difference between the ride there and back from various major cities to nearby suburbs, and to evaluate if our hypothesis holds true. We wanted to do this for 5 pairs of latitudes and longitudes (one smaller city/suburb for each of our major cities). We then wanted to compare the overall cost, based on each of our inputted latitude and longitudes, between Uber and Lyft. We wanted to compare our visuals for both Uber and Lyft to see the fluctuations in difference between the two companies. Lastly, we intended to show this in a third graphical representation that included data from both of the service's APIs.

2. The goals that were achieved.

After deciding that we would be determining whether the cost from a major city to a nearby suburb would be greater than the cost going back to the city from a nearby city, we were able to achieve most of our goals outlined above. Our idea to create a third visualization to compare overall average costs for Uber and Lyft was not able to be completed, mainly due to time constraints.

3. The problems that you faced.

In our initial stages of the project, we encountered issues with being able to access the Uber API. Because Uber had official API documentation for Python and we had to find unofficial Lyft documentation for python on GitHub, we thought Uber would be easier to deal with. However, we were easily able to access the API for Lyft, and after successfully creating our document with all of our code, attempted to replicate the call for Uber. The hiccup with Uber was minor, but required additional unanticipated time that we dedicated to getting our API to run.

Once we successfully called our APIs, we sought out to create our own latitude and longitude py file where we could select the start and end locations for every data point we would analyze. Again, Lyft was easier to work with than we had originally anticipated. Uber presented a minor obstacle again. Through debugging we realized that Uber would only allow latitude and longitude pairs within an 100 mile radius of each other (hence the 100 mile radius we mentioned above). This was an easy but tedious solution that required that we recalculate every single one of our location pairs (25 pairs of LL). We ran into minor syntax errors here (a lot of periods, commas, colons within a few lines), but quickly fixed them all.

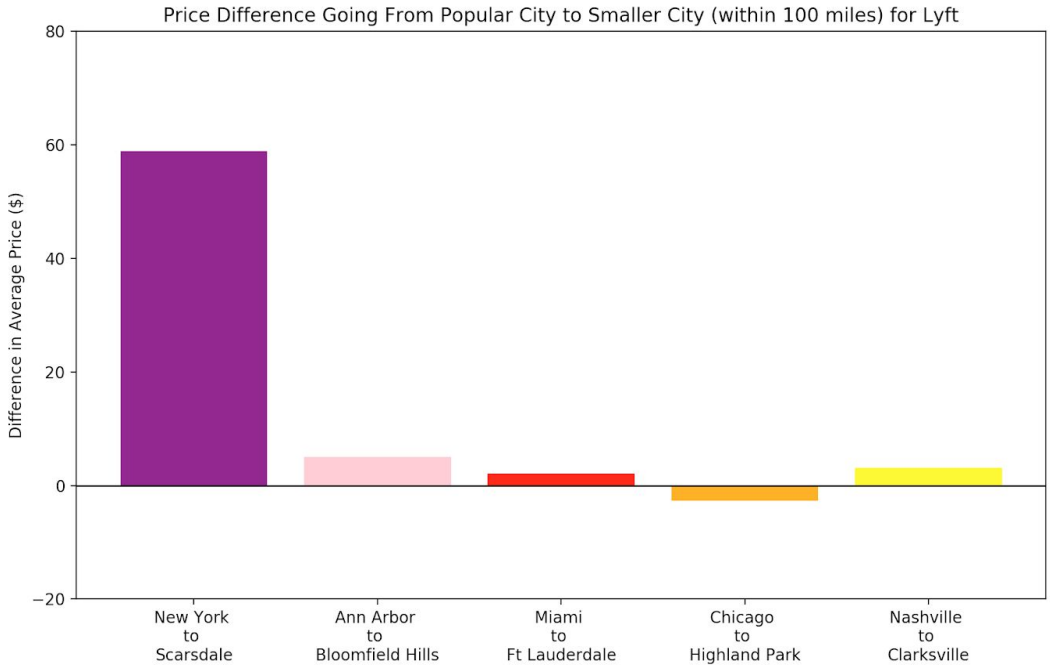
The largest, and most concerning problem that we encountered was when Dominique received an email regarding her Lyft API that stated that Lyft would be changing all of their public APIs to private within the 24 hours that the email was sent in. Thankfully we were already populating into our table correctly, so that night we inputted all of the information that we would need for the remainder of our project into the table (crisis averted).

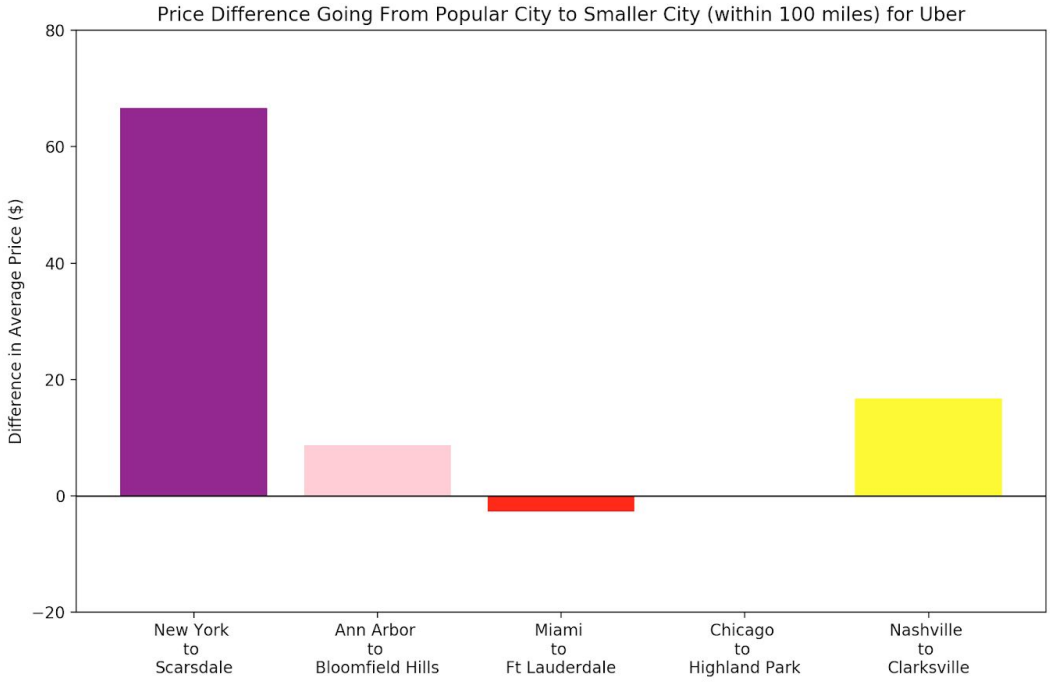
4. Your file that contains the calculations from the data in the database.

Name of file: writefile.csv

Cities	Average Cost There	Average Cost Back	Price Difference	Company
New York to Scarsdale	126.71428571428571	67.85714285714285	58.85714285714286	Lyft
Ann Arbor to Bloomfield Hills	65.71428571428572	60.714285714285715	5.000000000000007	Lyft
Miami to Ft Lauderdale	44.285714285714285	42.142857142857146	2.142857142857139	Lyft
Chicago to Highland Park	42.142857142857146	44.857142857142854	-2.714285714285708	Lyft
Nashville to Clarksville	63.142857142857146	60.0	3.142857142857146	Lyft
New York to Scarsdale	131.66666666666666	65.0	66.66666666666666	Uber
Ann Arbor to Bloomfield Hills	65.66666666666667	57.0	8.666666666666671	Uber
Miami to Ft Lauderdale	42.333333333333336	45.0	-2.6666666666666643	Uber
Chicago to Highland Park	43.0	43.0	0.0	Uber
Nashville to Clarksville	62.666666666666664	46.0	16.666666666666664	Uber

5. The visualizations that you created:





6. Instructions for running our code.

It is important to note that all of our files work regardless of the order your run them in.
We have populated a database with costs associated with the ride ordering platforms Lyft and Uber.

Running the info.py files:

1. Unzip [file name], and you will find 11 files.
2. Open the 2 info files (lyft_info.py and uber_info.py) in Visual Studio Code.
3. Open lat_long_list.py. Here, you will see the latitude and longitude pairs that we created for the purposes of our project. We have included a legend which specifies the name of the city/place associated with the latitudes and longitudes.
4. The Lyft and Uber APIs that we used required that we use wrappers, so install:
    - pip install uber-rides
    - pip install lyft_rides
    - pip install matplotlib (for use later on)

*If you will be populating our databases, it is important to note that Lyft changed their API policy from public to private throughout the process of our project. We were able to populate our table with all necessary information; however, the tokens given to us might no longer work, so you might not be able to populate information from Lyft if you attempt to run it.

The information that we have commented out should be left as is, so proceed to run the code normally in the terminal. The information that will appear in the database is as follows: the database is called rideshare and has two tables: RideShare (data from Lyft) and RideShareOtherCompany (data from Uber) with the following columns in each row:

- start_latitude
- start_longitude
- end_latitude
- end_longitude
- pair_id: each pair id has the same latitude and longitude but flipped. We were testing to see if the same car ride with the same two start and end locations had a difference in cost depending what city the car was ordered from.

    Example:

    Our first pair_id is for NY,NY and Scarsdale, NY. The first part of the pair id is the trip from NY to Scarsdale, and the second part of the pair id is the trip from Scarsdale to NY.
- costmin
- costmax

- time_requested: to record the various times at which we populated the database (in a real life situation, this would represent the time at which we called the Uber or Lyft)

* It should be noted that we limited the time between every time we ran the code to be 30 minutes, or (our set time) so that we could pop the database with pricing information from different times of day to be able to analyze and account for the fluctuation in price throughout a given day. This also prevented us from gathering repetitive data. Doing this enabled us then enabled us to take an average of the maximum cost for each ride. Due to this, if you were to run the code you would be able to populate into the database three times to get the information for each pair id we created in our lat_long_list. After three runs, the terminal returns that the information has been found in the database. You must wait 30 minutes before running and populating the database again.

Running the calculations.py files:

1. Open the 2 calculation files (lyft_calculations.py and uber_calculations.py) in Visual Studio Code.

* The information that we have commented out should be left as is, so proceed to run the code normally in the terminal. As noted above, you want to make sure matplotlib is installed so that you could see our visuals.

7. Documentation for each function that you wrote. This includes the input and output for each function.

File name: lyft_calculations.py

1. **Function:** get_costmax
   Input: id_num
   Output: 2 lists (EvenList and OddList)
   Documentation: Get the cost max for Lyft rides for each pair id of our select latitudes and longitudes. The passed in parameter (the input for the function) is pair_id from our database which maps to the column cost_max.This function returns two lists (the output for the function): EvenList is a list of the cost max for every ride from our select major city to a smaller one and OddList is a list of the cost max for every ride from the smaller city to that same major city.

2. **Function:** costmax_diff
   Input: EvenList, OddList
   Output: The calculated difference (a float)
   Documentation: The passed in parameters (the input for this function) are the EvenList and OddList returned from the previous function get_costmax. This function calculates two separate things. First, it calculates the average price for the EvenList (cost there: a

trip from a major city to smaller one) and the average price for the OddList(cost back: a trip from a smaller city to a larger one). The second thing it calculates is the difference in price for the two(cost there less cost back), to return the total difference in cost for a trip to-and-from the same locations (the output).

File name: uber_calculations.py

3. **Function:** get_costmax
   Input: id_num
   Output: 2 lists (EvenList and OddList)
   Documentation: Get the cost max for Uber rides for each pair id of our select latitudes and longitudes. The passed in parameter (the input for the function) is pair_id from our database which maps to the column cost_max. This function returns two lists (the output for the function): EvenList is a list of the cost max for every ride from our select major city to a smaller one. OddList is a list of the cost max for every ride from the smaller city to that same major city.

4. **Function:** costmax_diff
   Input: id_num
   Output: 2 lists (EvenList and OddList)
   Documentation: The passed in parameters (the input for this function) are the EvenList and OddList returned from the previous function get_costmax. This function calculates two separate things. First, it calculates the average price for the EvenList (cost there: a trip from a major city to smaller one) and the average price for the OddList(cost back: a trip from a smaller city to a larger one). The second thing it calculates is the difference in price for the two(cost there less cost back), to return the total difference in cost for a trip to-and-from the same locations (the output).

Although there are no functions for the lyft_info.py and uber_info.py files, we included a brief explanation of what is occurring within each file on lines 12-13 for lyft_info.py and lines 10-11 on uber_info.py.

8. You must also clearly document all resources you used. The documentation should be of the following form:

Date	Issue Description	Location of Resource	Result (did it solve the issue?

April 19, 2019	We were curious in seeing different ways that we could index in a for loop and ultimately get the index of a tuple within a list of tuples.	https://stackoverflow.com/questions/522563/accessing-the-index-in-for-loops	Yes, we understood how to index within a for loop and were able to iterate through the list of tuple to get the index of each tuple.
April 22, 2019	We wanted to review the syntax for writing a CSV file.	https://stackoverflow.com/questions/37289951/python-write-to-csv-line-by-line	Yes, this helped us figure out the syntax we needed.
April 24, 2019	After creating our CSV files, we wanted to add headers to the categorize the information.	https://stackoverflow.com/questions/28325622/python-csv-writing-headers-only-once	Yes, this helped us figure out how to add header columns.
April 15, 2019	We wanted to figure out how to add a timestamp to our database.	https://stackoverflow.com/questions/4363072/what-would-be-the-python-code-to-add-time-to-a-specific-timestamp	This helped guide us.
April 22, 2019	We needed to figure out how to set the values on our y-axis from -20 to 80.	https://stackoverflow.com/questions/3777861/setting-y-axis-limit-in-matplotlib	We were able to successfully set our y axis values with the information we found on this page. plt.lim(-20,80) so yes, success!
April 15, 2019	We needed to figure out how to store our datetime object as a dictionary to then be able to parse through it. We needed this so that we could set up the part of our code that would place a time limit between	https://stackoverflow.com/questions/36478257/datetime-object-in-jsonb-in-postgresql/36513900#36513900 https://stackoverflow.com/questions/41684991/datetime-strptime-2017-01-12t141206-	This helped us figure it out.

	every time we ran the code.	000-0500-y-m-dthms-fz	
April 3, 2019	Initial documentation to help us set up what we would need to access the Uber API from our computers.	https://developer.uber.com/docs/riders/ride-requests/tutorials/api/python	Yes, it worked.
April 3, 2019	Initial documentation to help us set up what we would need to access the Lyft API from our computers.	https://pypi.org/project/lyft_rides/	Yes, it worked.

The files contained within our zip submission are:
csv screenshot
graph_Lyft
graph_Uber
lat_long_list.py
lyft_calculations.py
lyft_info.py
uber_calculations.py
uber_info.py
rideshare.sqlite
lyft_tokens.py
uber_tokens.py
writefile.csv
README-Kidom-Final Report 206