As of August 7th 2019, “[Taxi] drivers spend 41% of their time in Congestion Zone cruising[[1]](#footnote-0).” Often, cab drivers (and even Uber drivers) will aimlessly drive until they get flagged down, or until someone needs them in a specific location. This inefficiency causes them to lose money, not properly account for supply in the high-demand parts of a city, and ultimately lead to too much taxi congestion if drivers don’t know how many other taxis there are in a given location. Cab drivers make on average $25 an hour[[2]](#footnote-1), but this is only for the hours that they have passengers. This means that if a driver works 10 hours, he is losing over $100 a day (given that he spends 40% of his day looking for passengers). This pattern of aimless driving not only results in financial losses for drivers, but it also hinders customers’ ability to locate taxis efficiently or secure rides at competitive prices.

Given all of these aforementioned struggles, a solution needed to be developed to help both taxi drivers and passengers. Even though Uber plays matchmaker between drivers and passengers, it does not give the proper information to cab drivers to make an educated decision to help maximize profit and minimize competition. To develop an application like this, there are a few essential steps that need to be done. First, there needs to be accurate and sufficient data to calculate which location would be best for a drive. Secondly, the data must be cleaned to remove the improper inputs that may have been inadvertently added. Afterwards, the data must be used to help create a proper regression model to predict useful information. Namely, how many cabs are in a given location, how much money would likely be produced in a certain taxi zone, how long the drives are, how far these drives are, or some combination of these factors is what would be useful. Once this is done, the model API must be deployed to the cloud where it can be called. Finally, a front end user interface will portray a map of the city that can highlight where “good” rides can be found by making a call to the backend.

When doing feature selection for modeling, it is important to determine which features best assist in predicting outcomes. Since the project is to allow taxi drivers to predict demand in live time in a particular area, the data had to be segmented geographically and temporally. This is a useful division to make since taxi demand at 2 AM in suburban Queens will look very different from Midtown demand at rush-hour time. New York City stores data as individual trips; as part of the ETL process, the data needed to be transformed by grouping it into geographical zones and dividing it up into 30 minute periods. Once the data is properly extracted and transformed, it becomes possible to perform EDA to see what factors contributed to demand. The first hypothesis was that historical data would be highly correlated with current data. Since it was expected that the day of the week would play a major factor in demand, it was unnecessary to look at the given date from one year and the same date from the previous year. Instead, every day was viewed based on the week of the year and the year of the month. February 1, 2018 becomes week 5 day 4, which correlates to February 2, 2017.

To test the hypothesis that historical data is correlated with current data, 2017 data and 2018 data on a random weekday in February were looked at. We found that in some neighborhoods, the demand is highly correlated. See Figure 1 as a snapshot of taxi demand in the Upper West Side in 2017 and 2018. Some neighborhoods have less of a correlation between historical and present data. In Figure 2, the Pearson correlation coefficient for a given day is mapped in every neighborhood. The Pearson correlation coefficient measures correlation on a scale of -1 to 1, where 0 is not correlated at all, and coefficients greater than .4 are significant. While not every neighborhood has a high correlation, most do. Historical data is the primary factor that was used to predict future demand.

| **UWS Trips 2017 vs. 2018 in Early February** |
| --- |
|  |
| **Figure 1** |

| **Pearson Correlation Coefficient by Neighborhood 2017 vs 2018** |
| --- |
|  |
| **Figure 2** |

The first model that was built was simply the historical data, which assumes that demand this year will be exactly the same as last year. As seen from Figures 1 and 2, this is a good intuition for a baseline model. To attempt to improve on that, a variety of machine learning models were made.

The first ML model was a simple DNN which took the taxi zone, time of day, and time of year as the input, and the number of trips, average duration, average price, and average distance as the output. This ultimately had two major problems. Because there are over 260 taxi zones, 52 weeks in a year, and 7 days in a week, a lot of data needed to be one-hot encoded. This led to a very sparse categorical data input space, which was prone to overfitting and very difficult to train with the computing power that was given. The second major problem with this approach is that of a vanishing gradient. When predicting taxi demand on the first Tuesday in April of 2024, the most important piece of context is the first Tuesday in April of 2023. With a standard DNN architecture, that important piece of context is not well retained within the weights and biases of the model.

To solve these problems, three additional models were built. An LSTM was created in an attempt to solve the vanishing gradient problem. Ultimately, this only yielded marginal improvements. The next attempt was to perform dimensionality reduction on the zone IDs. The zones were categorized into 30 categories, ranking from most popular to least. This improved the sparse categorical problem, but the vanishing gradient problem was still very much prevalent.

To further address this, the historical output data was incorporated into the input of the model. The EDA shed light on the fact that last year’s data of the same time and location is the most relevant context for predicting the current outputs. Since it was known what context was relevant, it was not necessary to ensure that it remained latent in the model. Relevant context can be fed as the input to the model for every prediction we were making.

| **Model** | **Mean Squared Error** |
| --- | --- |
| Baseline Historical | 4405 |
| DNN sparse | 2468 |
| LSTM | 1951 |
| DNN dimensionality reduction | 791 |
| DNN historical | 230 |

Ultimately, this last model produced the lowest MSE, and was the model that was incorporated into the application.

After the model was saved, a Flask application in Python was written with two endpoints. Each endpoint only requires one parameter, which is the date and time that one would like to get the information for the taxi zones (which, in our case, would be the current date and time - see below to see how it is formatted in a URL). The first endpoint, “/historical”, returns the information from the 2022 NYC taxi data, from the same week of the year and the day of the week of the date requested. The second endpoint, “/predictive”, returns the information predicted from the trained Machine Learning model. For both endpoints, the user is returned a JSON, which contains all the zones where there is information, including the average total price of the ride, the number of taxis present in that zone, the average duration of a ride from that zone, and the average distance of trips leaving from that zone (in miles).

The front end of this application uses Node.js with Typescript, using a Google Maps API to display the map for the user. The first step was to take the latitude and longitude information about the taxi zones[[3]](#footnote-2), and, using a Python script, convert them to be used as coordinates in Typescript for Google Maps polygons. Every 30 seconds, the Flask application gets called with the current date and time. The JSON then gets parsed and saved locally in the application.

There are multiple options for what the application displays, based on what the user clicks. The first option is to use the historical model (using the 2022 taxi information), or to use the predictive model (using the trained machine learning model). The zones are colored from red to green, with red being “bad”, green being “good”, and orange and yellow being somewhere in the middle. If there is no information on a given zone, it is colored gray. Within each of these models, there are five options on how to color the map. It can be colored by price of a ride leaving from that zone, duration of a ride leaving from that zone, distance of a ride leaving from that zone, the number of taxis in that zone, or a heuristic (which is calculated by Price/Duration\*Number of Taxis). Additionally, when any zone is clicked on, all of the information about that zone is listed based on which model is being used (including zone number, expected price, number of taxis, duration, and distance).

|  |
| --- |
| **Screenshot of the Application** |

During deployment, two instances of Google Cloud Compute Engine were utilized. One was dedicated to hosting the front-end components of the application, while the other served as the backend infrastructure. Once the machine learning model completed its training phase, it was seamlessly migrated into the cloud environment, enabling access from any device with internet connectivity. The accessibility of the model was simplified to require only the endpoint address and minimal user input. Users could specify their requirements, such as requesting historical or predictive data, by appending the appropriate parameters to the endpoint URL. For instance, accessing the predictive endpoint might resemble the following:

**34.16.191.120:8080/predictive?datetime=2024-02-29T16:44:00**

This approach facilitated efficient interaction with the model, providing users with timely insights and predictions based on their input criteria.

The second instance was needed to host the frontend. This is the code that contains the user interface which can actually show the map for the best taxi drives. Once the code was fully developed, it was also placed in a Google Compute Engine Instance that could be accessed from anywhere. The front-end would then call the back-end to produce the proper results and populate the map with the proper colors. Then, the user can decide which of the maps they would prefer to look at. Once this endpoint was up, the URL <https://bit.ly/TaxiHeatMap> was created to redirect to the endpoint. Once this was working, the user can get the full use of the application from anywhere in the world.

There were many challenges encountered throughout the process of making the application that had to be overcome in order for the model to be trained, the code to be put in the cloud, and the front-end to be properly formatted. The biggest problem for the model was the vast amount of data that needed to be used and lack of infrastructure and resources to benefit from all of it. In order to do proper EDA, cleaning, and training with all of the taxi data between 2017-2021 would have been impossible. To solve the EDA problem, a subset of the data was taken in order to create samples for what other months might look like, as well as to decipher which attributes were important. In order to clean the data, a query was made to only extract the most relevant and necessary data from the dataset. This way, it can all be used to help train the data. Finally, once the data was all received and cleaned, it took days in order to train a model that would produce proper results.

Another challenge that was encountered was having to learn how to make the front-end of this application, with most of the group having little to no front-end experience. While the syntax is easy to understand, it was challenging to write from scratch. Thanks to technological advancements of large language models (LLM’s), as well as ample documentation online, the application was able to be created. Additionally, learning to debug a front-end application was a learning experience. Google Chrome’s developer tools became very helpful, both in terms of seeing the HTML/CSS code, as well as seeing what was printed to the console.

Another difficulty that was dealt with was a lack of experience using the cloud. Getting the hang of cloud platforms and understanding their terminology was tricky. It took time to figure out which tools were appropriate to use and how to use them. Plus, concepts like efficiency, which types of compute engines to use, and memory size were new and needed thorough exploration. However, diving into tutorials and experimenting helped in getting comfortable with cloud technology and its deployment possibilities.

Some future improvements to this application would be to have the application take the driver’s current location to give them detailed information for the zone they’re currently in, instead of requiring the driver to know their exact location and click on it. Additionally, it would be beneficial to have the driver, who sees what’s going on in the zone that they’re in firsthand, give feedback to the application to help it update its model and make it better for taxi drivers using it in the future. Another beneficial feature would be for the driver to put in what they want to maximize on (most money, long rides, highest likelihood of getting a ride, etc.), and the application would tell them which zone to go to. An additional factor to consider is that if all taxi drivers use this application, then they would all gravitate towards the same zones. Those zones would then become extremely competitive, which would cause drivers to miss out on giving rides, as well as passengers in “weaker” zones not having any taxis come to them. A potential solution to this problem would be to give slightly different information to different users to avoid all drivers overloading to one zone. This would help both the drivers make more money, as well as passengers be assured there will be taxi drivers in their zone. While there are many improvements that can be made to this application, this will hopefully be a stepping stone for taxi drivers to use their working time more wisely, to have better rides for them, and to make more money.

1. https://www.nyc.gov/assets/tlc/downloads/pdf/proposed\_rules\_hvfhs\_cruising\_08\_07.pdf [↑](#footnote-ref-0)
2. https://www.indeed.com/career/taxi-driver/salaries/New-York--NY [↑](#footnote-ref-1)
3. https://data.cityofnewyork.us/api/views/755u-8jsi/rows.xml?accessType=DOWNLOAD [↑](#footnote-ref-2)