INFO 442 Data Science Projects
College of Computing & Informatics
Drexel University

TRAFFIC PREDICTION FOR NEW YORK CITY BASED ON GRAPH NEURAL NETWORK

FINAL REPORT

INSTRUCTOR
Professor Bhupesh Shetty

GROUP NUMBER
Group 2

GROUP MEMBER
Harry Zhao
Yantian Ding
Yi Pan

DREXEL UNIVERSITY
College of
Computing &
Informatics

ABSTRACT

The real-time speed of each roadway segment in a network is essential in many urban traffic environments, from transportation route planning in freight transportation to congestion avoidance in advanced traveler information systems. Therefore, accurate real-time average road speed prediction is of utmost importance. Vehicle speeds and travel times can vary considerably in urban environments due to traffic congestion caused by accidents or adverse weather conditions. At another level, one can also predict the future of each road segment by the speed variation of the surrounding road segments and the impact of the surrounding road segments on that road segment.

This paper presents a project to predict the real-time average speed of selected road segments in New York City using graph neural networks. It focuses on predicting road segment speeds using Movement data from Uber vehicles and road network data collected over a considerable period as input. In this paper, different machine learning and data mining methods are applied, and computational results are reported on actual data to demonstrate the accuracy of the obtained predictions and the model's superiority.

Keywords

Deep Learning
Prediction
Traffic Speed
Graph Neural Network
Time-series Data

Traffic Prediction for New York City based on Graph Neural Network

Harry Zhao, Yi Pan, Yantian Ding
College of Computing and Informatics
Drexel University
Philadelphia, the United States

***Abstract*—The real-time speed of each roadway segment in a network is essential in many urban traffic environments, from transportation route planning in freight transportation to congestion avoidance in advanced traveler information systems. Therefore, accurate real-time average road speed prediction is of utmost importance. Vehicle speeds and travel times can vary considerably in urban environments due to traffic congestion caused by accidents or adverse weather conditions. At another level, one can also predict the future of each road segment by the speed variation of the surrounding road segments and the impact of the surrounding road segments on that road segment. This paper presents a project to predict the real-time average speed of selected road segments in New York City using graph neural networks. It focuses on predicting road segment speeds using Movement data from Uber vehicles and road network data collected over a considerable period as input. In this paper, different machine learning and data mining methods are applied, and computational results are reported on actual data to demonstrate the accuracy of the obtained predictions and the model's superiority.**

Keywords-component; Deep Learning; Prediction; Traffic Speed; Graph Neural Network; Time-series Data

I. INTRODUCTION

Travel speed prediction is a significant problem in most urban traffic models, regardless of whether they involve the transport of people or goods. Most travel speed variations and travel time variations are caused by traffic congestion, which can be classified as recurrent and non-recurrent, i.e., due to accidents, construction, emergencies, special events, and bad weather. Therefore, it is necessary to accurately predict the travel speed (time) under recurrent and non-recurrent congestion. Notably, better forecasting may greatly assist individuals and transportation companies operating in urban areas to plan their activities and may lead to significant reductions in actual travel times and greenhouse gas emissions.

As the amount of available data collected from detection vehicles, smartphone apps, and other location technologies continues to grow, the challenge of travel speed prediction is no longer related to the amount of data but instead to modeling and extracting useful information from this data. In this context, models and algorithms based on actual data have great potential to accurately predict travel speeds on different types of streets and roads at different times of the day.

The project presented in this paper is one of the final projects of the DSCI 442 Data Science Project course at Drexel University. The project aims to predict travel speeds on significant roadways in New York City to avoid congestion through real-time prediction of speeds at each location. This paper aims to make the best possible driving speed predictions using speed and geolocation data collected from the Uber APP on the Uber Driver's phone. The data used in this study comes from Uber Technologies Inc, a company that develops mobile applications to connect riders and drivers in a sharing economy service that provides passenger vehicle rentals and matchmaking rideshare [1]. Uber provides access to their data on their platform Uber Movement, and their purpose is to "provide data and tools for cities to more deeply understand and address urban transportation challenges."[2]

From a methodological point of view, our problem-solving approach is composed of several algorithmic components that implement the following macro-steps:

- data preparation and cleaning
- exploratory data analysis
- model building and training
- Prediction and Evaluation

In the context of velocity prediction, the literature addressing these macro-steps is not uniform, leading to many possible algorithmic choices. Therefore, special attention must be paid to the context and to our specific data when determining the best choice. In other words, our contribution is tailored to the data: we must solve the problem for the company providing us with the data, just like any other data-driven predictive or prescriptive application. While the algorithmic components are not truly novel, the contribution of this paper lies in their effective combination to create an approach, as defined by the four macro-steps above, that is general to velocity prediction and works well on our dataset. The components of the algorithm that produce the best performance may be different for different datasets, but this does not affect the methodology itself. We expect other researchers and practitioners to follow such a methodology and use additional macro steps if their datasets require it.

The rest of the paper is organized as follows. We first review the analysis of our problem itself in Section 2. Then, Section 3 describes the data we used and the process of extracting the required data from the Uber Movement and OSM data. Then, an exploratory analysis of the data is performed in Section 4 to understand the data's insight further. Subsequently, Section 5 describes the neural network model used to predict vehicle driving and reports the computational results in real-world situations. Finally, Section 6 compares our model with other solutions and analyzes the advantages and disadvantages of our model.

II. PROBLEM

A. *Problem Statement*

The economic construction and development of cities are closely related to the road conditions of cities, and good road access is conducive to promoting urban economic growth. For a cosmopolitan city like New York, the massive population will inevitably lead to traffic congestion, which greatly wastes people's time resources.

At the same time, traffic congestion can have different impacts on both companies and individuals. For individuals, traffic forecasts can primarily affect the efficiency of personal travel, failing personal travel plans to proceed smoothly. At the same time, traffic congestion can lead to a significant waste of personal time resources, allowing people to waste their precious time in endless and meaningless waiting. For companies, traffic congestion is crucial for many road transportation-related industries. For these companies, their profitability depends heavily on the traffic conditions on the road ahead. For example, fast traffic accessibility can increase the profitability of freight companies, or excellent traffic planning can help navigation system-related companies to rise quickly in the market.

Traffic congestion is divided into recurrent and non-recurrent, and to be able to predict both kinds of congestion, we choose to predict congestion further by predicting the real-time speed of each lot.

B. *Benefit*

Traffic congestion prediction is essential to improve the efficiency of urban traffic operations. For city residents, traffic congestion prediction can provide a basis for planning travel time and travel routes for city residents to maximize travel efficiency. Secondly, it can help the delivery staff to improve the efficiency of delivery to ensure the freshness of food, just like DoorDash, which is one of the leading delivery companies in the U.S. due to its excellent route planning system to ensure the food can be delivered to the customers quickly to the maximum extent. At the same time, it can help traffic authorities to plan and direct traffic, thus avoiding traffic congestion. In addition, traffic managers can also explore traffic congestion places. Combined with the actual road conditions and urban road facilities, traffic signs and guidance, traffic marking settings, and other work, existing urban roads resources are reasonable to solve traffic congestion on the urban road network.

III. DATA

A. *Data Description*

1. Uber Movement

Every hour of every day, people are using Uber to travel through more than 450 cities worldwide. From Sydney to New York, Uber has been working to understand these cities to make them cleaner, more efficient, and less congested. Along the way, Uber has found local leaders, urban planners, and civic communities working to decipher their cities' commuting problems and figure out the best ways to invest in new infrastructure. [3]

In January 2017, Uber released the "Uber Campaign," a tool for urban planners and researchers to study improving urban transportation. Specifically, the dataset includes more than 2 billion Uber trips in cities such as Bogota-, Boston, Johannesburg, Manila, Paris, Sydney, and Washington, D.C.. It includes the arithmetic mean, geometric mean, and standard deviation of travel times aggregated over a selected date range between each pair of zones1 in these cities. Uber Movement is available to the public and can be downloaded directly from the [Uber Movement website] to download in .csv format.

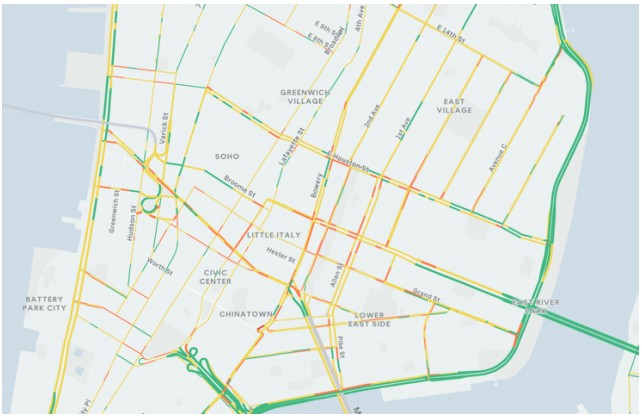

Figure 1. The Uber Web interface colors roads in the New York city based on the average travel speed

In our project, we chose to use the January 2020 data as our dataset. In this dataset, velocity data in hourly units for 31 days and 24 hours are included. The information contained in this data is shown in Table 1.

TABLE I. ATTRIBUTES IN UBER MOVEMENT DATA

Attribute	Description	Data Type
Year	Year(Local city time)	**Int**
Month	Month 1-12(local city time)	Int
Day	Day 1-31(local city time)	Int
Hour	Hour 0-23(local city time)	Int
Utc_timestamp	Unix time for start of hour (UTC)	Int
Segment_id	Movement ID that maps to a specific road segment.	Text

start_junction_id	Movement ID that maps to start intersection of traversal.	Text
end_junction_id	Movement ID that maps to end intersection of traversal.	Text
Osm_way_id	Corresponding OpenStreetMap Way ID for this segment.	Bigint
osm_start_node_id	Corresponding OpenStreetMap Node ID for this junction.	Bigint
osm_end_node_id	Corresponding OpenStreetMap Node ID for this junction.	Bigint
speed_mph_mean	Average speed of Uber vehicles on this road segment in mph	Float
speed_mph_stdev	Standard deviation of speeds on this road segment in mph.	Float

2. OpenStreetMap

OpenStreetMap is a collaborative project for building free content online maps to create a world map with free content that all can edit and easy navigation options for available mobile devices. [4]

The data format used by OSM is a topographic data structure, which consists of four core elements [5].

- **node**: stores the latitude and longitude, indicating the location, but not the actual size of the node on the map, such as an attraction or a mountain, or a store or a restaurant, or as part of a path. Nodes can be attached to paths and relationships.
- **way**: An ordered arrangement of nodes, presented as a dash, can also loop back to the starting node to form a closed path and can be presented as a circular path or as a polygonal area. This source material can be used to present linear material, such as a street, river, etc., or a polygonal area, such as farmland, a park, a parking lot, a building, a campus, or a lake or a lake forest. Paths must have nodes displayed on the map and can be attached to relationships. Path information can be calculated as the length, area, or perimeter of a polygon.
- **Relation**: Nodes, paths, and relations are ordered (the three types of primitives are collectively referred to here as "members"), where each member optionally has a "role" (string) that determines the nature of that member in the relation. Relationships are used to represent the relationship of individual primitives (nodes, paths, and relationships), such as the turn restrictions of a road, a boundary formed by different paths, a national, provincial, or railroad route, or multiple polygons in the middle of a region (e.g., an atrium surrounded by a circular building, or an island in a body of water), where a "role" string can be used to describe the relationship between them.
- **tag**: A key-value pair (all keys are strings) that stores the metadata (type, name, and physical characteristics of the object) of the objects on the map, giving meaning to the OSM information and representing the existence of something in the real world, and information about it. Tags cannot exist independently. They must be attached to an existing object, i.e., a node, path, or relationship.

With the data provided by Uber, we can obtain the OSM IDs corresponding to each road segment's start and endpoints, so we can build a graph to model the basic urban structure in which the trip occurs. Using the data provided by Uber Movement, we can quickly build a basic city structure on OpenStreetMap.

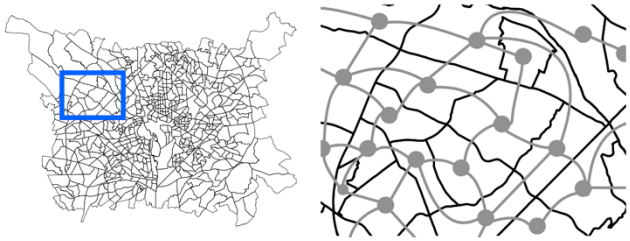

Figure 2. The OSM System contains a set of polygons on which we'll zoom in (left). Looking closely at the region in the blue rectangle, we define a graph where every node represents a region and two regions are linked if they are adjacent (right)[6]

B. *Preprocessing*

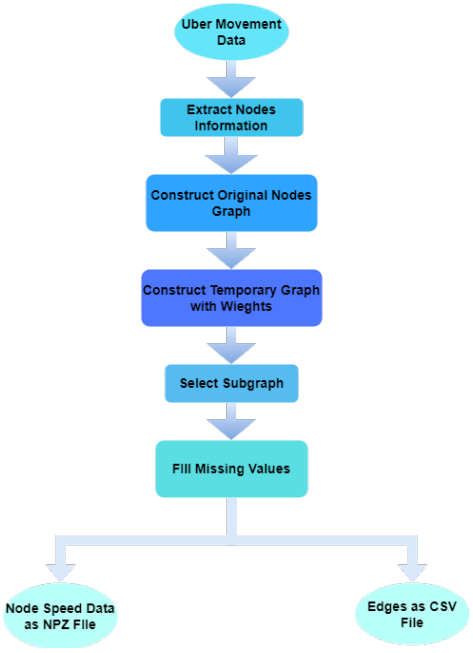

Figure 3. Workflow of our Preprocess

In this project, we will use uber movement speed data from New York City to try to build a prediction model for the average speed and congestion of road segments using a graph neural network approach. This is the flow chart of the whole pre-processing process.

1. Raw Data

We used pandas to read the data.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	year	month	day	hour	utc_timest	segment_i	start_junct	end_junct	osm_way_	osm_start	osm_end_	speed_mp	speed_mph_stddev	
2	2020	1	11	12	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	23.97	5.266	
3	2020	1	10	16	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	25.061	3.764	
4	2020	1	21	16	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	24.384	1.354	
5	2020	1	24	8	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	23.314	2.604	
6	2020	1	17	9	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	24.605	4.452	
7	2020	1	31	13	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	21.012	2.812	
8	2020	1	20	22	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	23.826	1.846	
9	2020	1	25	15	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	14.267	3.275	
10	2020	1	24	17	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	23.573	2.975	
11	2020	1	11	18	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	21.619	4.092	
12	2020	1	28	8	2020-01-:	ffd5b0eb2	33022b4bf	61e031cf1	6.28E+08	5.92E+09	5.92E+09	24.929	3.335	

Figure 4. Raw Data

2. Extract nodes information

a. Extract valid data from the original data ("day", "hour", "osm_start_node_id", "osm_end_node_id", "speed_mph_mean", "speed_mph_stddev").

b. Replace the data - replace the tuple of "osm_start_node_id" and "osm_end_node_id" with " node_id" and create a replacement table.

3. Construct a temporary graph with the number of records as weights

Calculate the amount of data for each edge. If there is a record for a road in that hour, we will add one to that edge. We call this value "weight" and attach it to the node replacement table.

	A	B	C	D	E	F
1	road_id	start_node_id	end_node_id	weight		
2	1	5924962522	5924962521	67		
3	2	42856931	5924962521	166		
4	3	2300353500	5212739096	628		
5	4	3838008669	5212739096	504		
6	5	5487813168	5212739096	84		
7	6	42750747	42747630	38		
8	7	42855147	2883544414	100		
9	8	60922878	595757808	668		
10	9	42489091	42486694	408		
11	10	42486693	42486694	27		
12	11	42496752	42486694	359		
13	12	42449139	42435894	39		
14	13	42435898	42435894	463		
15	14	42435891	42435894	404		

Figure 5. Node Replacement Table (with weight)

4. Select subgraphs according to conditions

Extract the sub images we need from the main image. We have used a ninety-five percent confidence interval for the edge weights based on statistical experience. Here the edge weights represent the number of data records per road segment in the month of January. Since the recording is done in hours, one day should have 31*24=744 pieces data. We consider the real data within the ninety-five percent confidence interval to be significant, and our assumption of fill for the exact value does not affect the distribution of the data itself, so we choose the road segments with less than five percent record fill as our data set. That is, road segments with a true data count of 706.8(we use 707 in the code) or more will be included in the sub graph we eventually use.

	A	B	C	D	E	F
1	source	target	road_id	weight		
2	3530981376	4202517943	24785	556		
3	3530981376	3530981377	17767	557		
4	3530981377	4202418640	42750	563		
5	3530981379	4202418642	16443	542		
6	3530981379	4202418640	50404	566		
7	3530981380	4202418642	76608	539		
8	3530981380	4202418643	82651	551		
9	3530981381	4202418643	14045	549		
10	3530981381	3530981382	7851	555		
11	3530981382	4202418645	27706	553		
12	3530981383	4202418645	10070	552		
13	3530981383	4202418647	40620	555		
14	2375712773	42468921	31711	664		
15	2375712773	42468925	99455	666		

Figure 6. Subgraph Edge List (with weight)

5. Fiiling in missing values

Here we merge the extracted edge weights csv file with the corresponding velocity csv file for each edge. Also, we fill the speed records for all the missing edges. Since the speed limit in the New York State metropolitan area is 25miles which is about 40 km/h. Therefore we populated the time periods of the missing records assuming that their speed records are 40km/h[7]. According to data statistics of New York Taxi and Limousine Commission, we found that in January 2020, the point in time to which our data belong, an average of about 470,404 Uber trips were generated in New York City each day. With such a large uber usage rate, we think it is safe to assume that when an area is not covered by uber records, the area is not busy at that time. In other words, the roadway is likely not congested at that time of day. Also, we assume that people drive to follow the speed limit but as fast as possible, so we fill in the missing speed data using the vehicle speed limit for New York City.

	A	B	C	D	E	F	G
1	day	hour	road_id	speed_mph_mean	spped_mph_stddev		
2	1	0	24785	23.231	7.068		
3	1	0	17767	27.017	4.603		
4	1	0	42750	26.704	4.395		
5	1	0	16443	21.793	8.823		
6	1	0	50404	25.798	6.274		
7	1	0	76608	14.552	5.43		
8	1	0	82651	8.695	4.943		
9	1	0	14045	9.608	4.873		
10	1	0	7851	10.728	8.548		
11	1	0	27706	11.367	8.948		
12	1	0	10070	17.135	11.053		
13	1	0	40620	21.199	7.987		
14	1	0	31711	19.59	8.499		
15	1	0	99455	16.773	7.61		

Figure 7. Uber Movement Data

6. Store the relevant data for each day as well as the node speed data as NPZ file, and the list of edges of the finalized eligible subgraphs is CSV form

For the model, we store the data information of each node of the subgraph as an NPZ file and the list of edges as a CSV file.

	A	B	C	D	E
1	from	to			
2	73	5			
3	5	154			
4	154	263			
5	263	56			
6	56	96			
7	96	42			
8	42	58			
9	58	95			
10	95	72			
11	72	271			
12	271	68			
13	134	107			
14	107	130			
15	130	129			

Figure 8. Nodes.csv

IV. EXPLORATORY DATA ANALYSIS

The overall process of our Exploratory Data Analysis is presented roughly, as shown in the figure below.

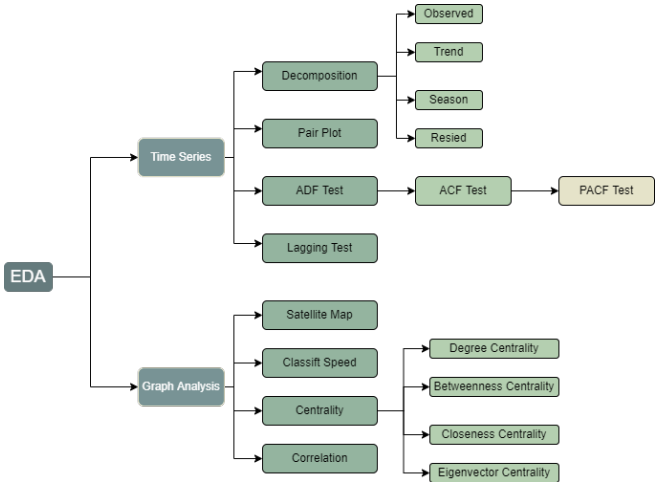

Figure 9. EDA Steps

Our exploratory data analysis is carried out in two main aspects. One aspect is to analyze the data in terms of time series to explore the distribution of the data over time. On the other side, we explore the data by using graphs to determine corresponding features in the network results. The reason for dividing the EDA into two parts is that our data is road data with a network structure. Based on this network data, we collect information about the speed of vehicles and the time during a month, so we divide the exploration process into two parts: temporal and spatia

*A. **Time Series Data***

We analyze the data from four aspects regarding the Time Series: Decomposition, Pair Plot, ADF Test and Lagging Test.

1. Decomposition

In Decomposition, we investigate whether the data has a cyclical pattern of change. We divided Decomposition into four small phases for analysis. We first drew four decomposition graphs to analyze the time-series relationship of each hour of the road network. As shown in the figure below

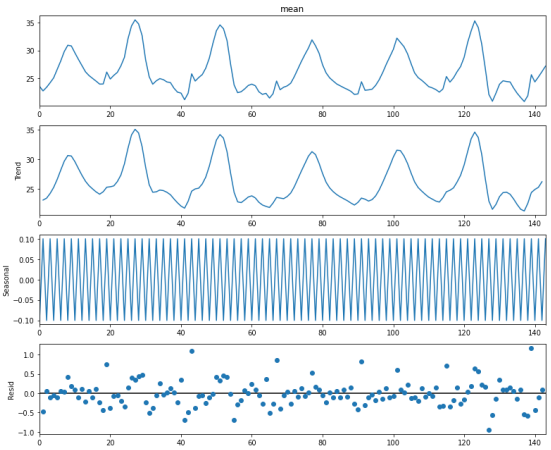

Figure 10. Hourly Speed Decomposition

The first graph is a line graph of the actual observed hourly average speed. We use the line graph to reflect the trend of the data. Here is a smooth trend, with periodic increases or decreases over time, with no unusual changes occurring. The second graph shows the overall trend change data. From the graph, we can see that the trend of the average speed data is smooth, with no significant increase or decrease. The third graph shows the seasonal data exhibited by the whole series. By looking at the images, we see that the average velocity data varies periodically with a steady frequency. Therefore, we consider our time-series data to be seasonal. The fourth graph is a plot of the residual data. The residuals represent the random variation in the time series, which we must focus on predicting. By looking at these images, we find that the residual values of the average velocity data are regularly distributed on both sides of 0, indicating that a linear correlation may emerge. We will continue to explore the possibility of the linear correlation aspect later.

Similarly, we also apply the four decomposition plots to the daily average speed data for all road segments to explore the temporal relationship. This is shown in the following combined plot.

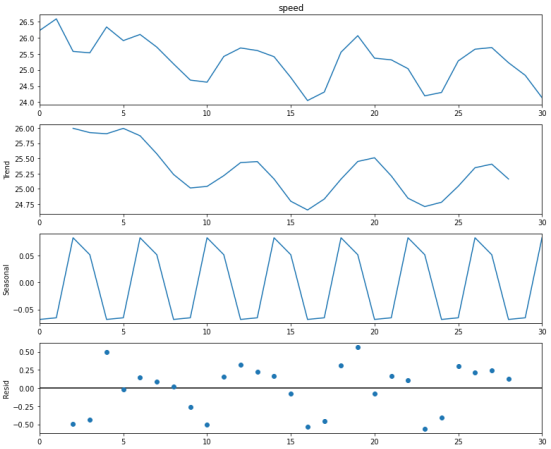

Figure 11. Daily Speed Decomposition

The first graph is a line graph of the actual observed daily average velocity. We used the line graph to reflect the curve

of the data. You can see that there is a downward trend here, indicating that people's driving speed showed a downward trend in January 2020, probably due to weather, such as cold weather with snow causing icy roadsides or snow blocking roads, etc. If weather data is available, we can compare it for deeper exploration. The second graph shows the overall trend change data. From the graph, we can find that the trend of the average speed data is smooth, offering a gradual decline, which fits with the first graph. The third graph shows the seasonal data exhibited by the whole series. By looking at the images, we find that the average velocity data varies periodically with a steady frequency. Therefore, we consider our time-series data to be seasonal. Every so often, the velocity will show the same trend, for example, every week, with the Monday of each week showing the same velocity. The fourth graph is a plot of the residual data. The residuals represent the random variation in the time series, which we must focus on predicting. By looking at the images, we find that the residual values of the average velocity data are regularly distributed on both sides of 0, indicating that there may be a linear correlation in velocity as well.

Besides, we also apply the decomposition on the daily number of travel records to see a corresponding time series relationship. This is shown in the following combined graph.

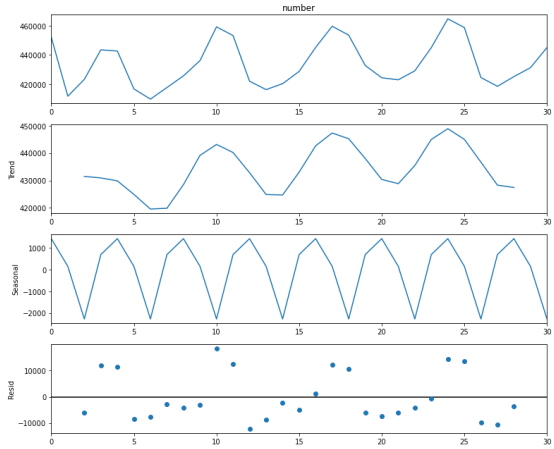

Figure 12. Daily Records Number Decomposition

The first graph is a line graph of the actual values we observed. We use the line graph to reflect the trend of the data. We can see an upward trend, indicating that more and more uber vehicle trip data records are collected as January progresses, showing an opposite change to the decrease in average daily speed. The second graph shows the overall trend change data. From the graph, we can see that the trend of the transportation data is smooth and offers a gradually increasing trend of change. The third graph shows the seasonal data exhibited by the whole series. We find that our data vary periodically with a steady frequency by looking at the pictures. Therefore, we consider our time-series data to be seasonal; for example, more recorded data may be collected weekly on Mondays than Fridays. The fourth graph is a plot of the residual data. The residuals represent the random variation in the time series, which we need to focus on predicting. By looking at the images, we find that the residual values of the data are regularly distributed on both sides of 0, as analyzed in the previous two graphs, indicating that there may be a corresponding linear relationship here as well.

2. Correlation Analysis

We perform the correlation analysis of the pair plot after decomposition. As shown in the following pair plot.

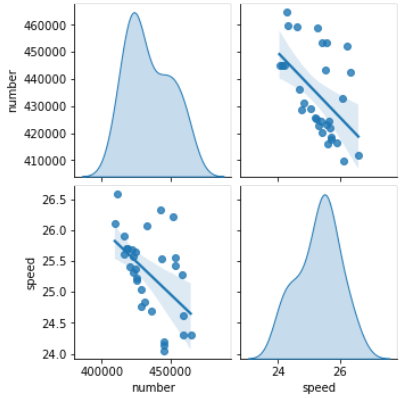

Figure 13. Corrrelation Pair Plot

We use a pair plot to show the relationship between the two variables. Here we put the daily collected vehicle data records together with the daily average speed to compare and plot the pair plot as shown above. Figures 1 and 4 are the corresponding linear plots indicating the concentrated distribution of the data, which shows that the overall distribution of the collected daily uber data volume is between 40,000 and 48,000. In contrast, the daily average speed is mainly distributed between 24 and 26. Figure 2 and Figure 3 put the two features together to draw a scatter plot. We can see a negative linear correlation here. We explain it like when the amount of data collected increases, the average speed decreases, while the average speed increases when less data is collected. This is a reasonable representation because an increase in the amount of data collected means that more vehicles are on the road when traffic congestion is more likely to occur due to the rise in vehicles, thus reducing the overall average speed. Here in combination with the performance of the previous decomposition data, we get the exciting discovery that there is a negative linear correlation between the amount of uber driver data collected daily and the average speed.

After that, we did the ADF test on our series data. Through the ADF test, we calculated that the p-value of our data is 0.008689, which is less than the confidence level of 0.05. The premise of the ADF test is to assume the existence of a unit root. By calculating the p-value, we compare the p-value with the confidence level to decide whether to reject the original hypothesis. If the value p-value is less than the confidence level, we reject the original hypothesis, indicating that the initial hypothesis is wrong and there is no unit root. The existence of no unit root means that the series is smooth and does not have randomness. Whereas, if the value p-value is

more significant than the confidence level, the original hypothesis cannot be rejected, indicating that the data is not smooth and random. Here we have a confidence level of p-value less than 0.05. A confidence level with 0.05 is often taken according to the rule of the thumb, which indicates that our data is stable and has no unit root, which means our information is not randomly selected. This also ensures that we can proceed with the ACF Test and PACF Test.

Since we have illustrated that our data is stable by ADF Test, we are pleased to perform ACF Test. as shown in the figure below.

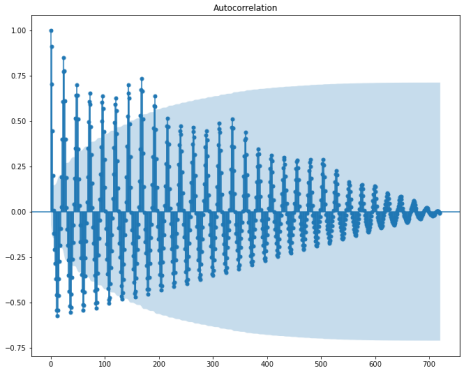

Figure 14. ACF Diagram

Here we use ACF Test to test the stability of the time series. We plot the Autocorrelation image as shown above. For a stable time, series, autocorrelation means that it is possible to predict the future using historical data. The blue area is the error range. The data in this region are considered insignificant. The pattern of decreasing ACF amplitude generally implies the presence of autocorrelation, as we have found and demonstrated in our previous time series exploration. From the figure, we can see that the ACF is stable as its values gradually converge and approach zero, thus demonstrating the stability of the time series, which means the series is of no randomness. Because both ADF Test and ACF Test illustrate the strength of the data, we can continue to explore deeper and analyze our data in terms of PACF.

We plotted the partial correlation image as shown in the figure below.

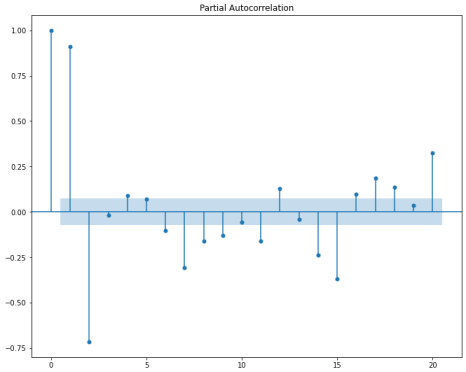

Figure 15. PACF Diagram

PACF considers only cycle-specific correlations, which gives us a good starting point for determining the autocorrelation cycle, so in PCAF, we can find which processes are autocorrelation. In the ACF plot, the ACF is a trailing tail that gradually tends to 0. In PACF, it is very close to 0 when the lag is 3, so we can consider using a model like ARMA (3,0) to form a time series model to use past data to make predictions about possible future data.

After completing the above stability test, our time series data are smooth and not randomly distributed but show regular variation, consistent with the decomposition phenomenon. At the end of the time series exploration, we performed a lagging test, which compares the current data after a delay period to determine whether they have a corresponding relationship. As shown in the figure below, we set the delays as 1, 2, 3, and 4hour to analyze the relationship between the two.

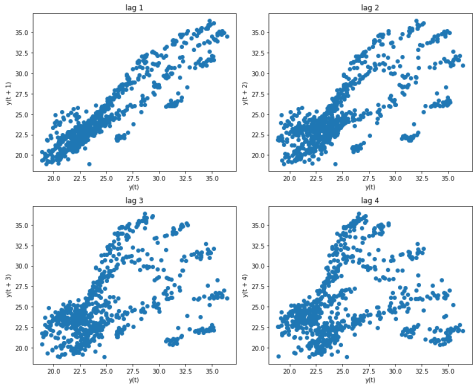

Figure 16. Lagging Test

As shown in the figure, when lag is 1, we can find a clear positive linear correlation, and as the lag value gets larger, the divergence of the scatter plot increases. The linear correlation decreases, but it still exists. Because we use the hourly speed data of road network sections, our data has both temporal and spatial attributes. When congestion occurs on a road section, the road sections around this road section are likely to receive the impact of that road section and experience congestion. And as time goes by, this congestion will gradually disappear, so when the lag value is small, indicating that it is not far from the current data time, the two will show a very, very strong linear correlation, as shown in the lag1 figure, the scattered points almost overlap to form a thick line. And as the lag value increases, as time passes, the two gradually diverge, and the linear correlation gradually decreases but still has some degree of linear correlation. This is a significant finding; the performance of the data will show a positive linear correlation within a specific time difference, while the correlation gradually decreases until it disappears as time goes on

B. ***Graph Data***

1. Data Visualization

By using the JOSM editor, we have generated the respective network on an accurate New York City map, like the following:

Figure 17. Graph Data Visulization

In the above figure, the highlighted road segments are the ones we have selected. The selected road segments are mainly located in Manhattan, Brooklyn, and Queens, most of which are in the busy areas of New York and have a high practical significance. At the same time, we also observed that some road sections in our selected subplot are in remote locations, which are some distance away from the busy areas of New York. These road segments may be the relative outliers of our prediction task in this project and the data itself.

2. Congestion Level Division

Because we need to predict traffic congestion based on speed. We calculate the specific banner value among different classes According to Level of Service (LOS) with Average Speed and Traffic Conditions for Arterial Roads [8]. We divide into four zones according to different speeds as follows.

- [33.80259, ∞): Unblocked
- [20.87807, 33.80259): Mild Congestion
- [15.9071, 20.87807): Moderate Congestion
- [0, 15.9071): Serious Congestion

The histogram of these four road sections as a percentage of the total number of road sections is shown below. We can see that congestion accounts for the largest share of them, over 10,000 out of about 21,000 data. The bar chart shows that New York City is at least mildly congested in most cases.

Figure 18. Histogram of Congestion Level Distribution

3. Centralirity

a. Degree Centrality

Firstly， it comes to Degree Centrality. Degree centrality is the most direct metric for portraying node centrality in network analysis. A more considerable degree value of a node means a higher degree of centrality of the entire node, and the more influential the node is in the whole network. According to the definition given by Newman (2010), in this project, the formula of degree centrality is:

$$DegreeCentrality(\mathrm{j}) = \sum_{j=1}^{n} \mathrm{A_{ij}}$$

For the computed Degree Centrality, we plotted the bar chart shown below.

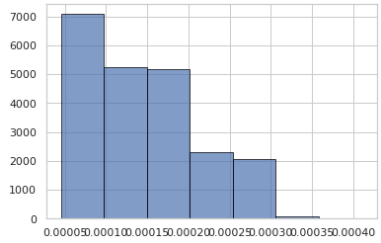

Figure 19. Degree Centrality Distribution

This figure shows that the degree centrality of our selected waypoints is concentrated in the interval of 0-0.0003. A single point, alone, is not essential in the whole road network we have selected.

We also plotted a categorical scatter plot, as shown below.

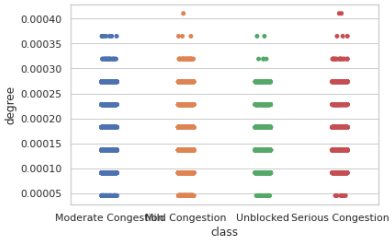

Figure 20. Degree Centrality Categorical Scatter

This figure shows that the degree centrality in all four classes is distributed uniformly but discontinuously. The degree centrality value is all small cause we have selected more than 20000 points, and the highest degree of centrality for a single point is 9.

Since we have classified the congestion level into four categories based on speed, here we combine these four categories with the degree centrality to draw the combined graph shown below.

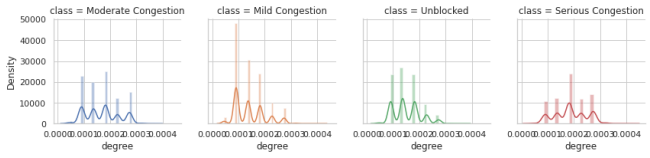

Figure 21. Degree Centrality Distribution

This figure shows that the degree centrality of the four classes is discrete in the interval of 0-0.0004. Meanwhile, most points are concentrated in each graph in 5 bins.

We draw the box diagram below at the end of the degree centrality

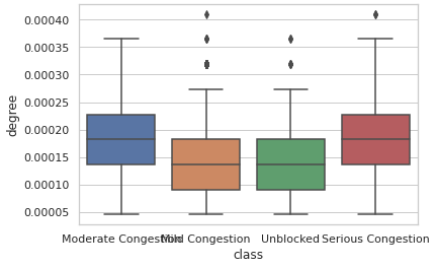

Figure 22. Degree Centrality Boxplot

This figure shows that the mean degree centrality is higher under Moderate Congestion and Serious Congestion, while lower under Mild Congestion and Unlocked. Similarly, the range of data under Moderate Congestion and Serious Congestion is more extensive than that of Mild Congestion and Unblocked.

The degree centrality graph shows that nodes with high degree centrality tend to be moderately and heavily congested roads; nodes with low degree centrality tend to be lightly congested and clear roads. This is consistent with our general intuition that intersections tend to be more crowded than non-intersections.

b. Betweenness Centrality

After analyzing the degree centrality, we analyze the betweenness centrality. Betweenness centrality describes the ratio of the number of times a node acts as a bridge for the shortest path between two other nodes to the number of short paths in the network graph. According to the definition given by Everett and Borgatti (1999), Betweenness centrality is calculated as:

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

First, we plot the histogram of betweenness to observe the overall distribution, as shown in the figure below.

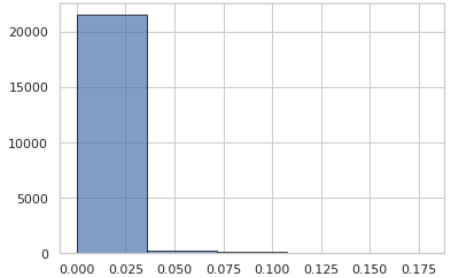

Figure 23. Betweenness Centrality Distribution

This figure shows that the betweenness centrality of our selected nodes is low. It is concentrated in the range of 0-0.025

After that, we plotted the categorical scatter plot, as shown below.

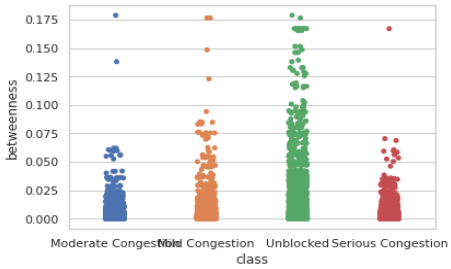

Figure 24. Betweenness Centrality Categorical Scatter

This figure shows that the Betweenness Centrality of all points in each class is concentrated at a low value. However, except for the Unblocked class, the shape of the distribution of all types is more "extreme" (when betweenness Centrality >0.1, the distribution is sparse for all three courses except for the Unblocked class). And as the congestion level increases, the distribution of betweenness centrality becomes more concentrated.

As shown in the figure below, we plot the corresponding density of betweenness according to the different congestion levels.

Figure 25. Betweenness Centrality Distribution

This figure shows that the density trends of the four-velocity regions are roughly the same, and the data are concentrated in the interval of 0-0.05. Still, the distribution of betweenness centrality of Unblocked is flatter, and the distribution of the remaining three categories is more concentrated.

Finally, the box plot is performed for betweenness centrality

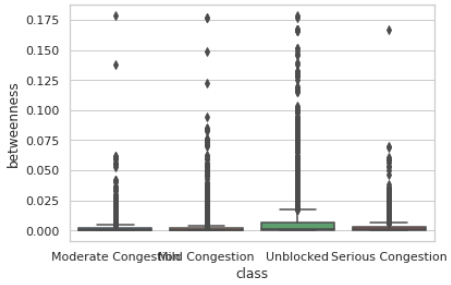

Figure 26. Betweenness Centrality Boxplot

This figure shows that the upper limit of betweenness centrality in all the classes is close to 0, except for the unblocked type, which has the highest upper limit value at about 0.02. And, the unblocked course has a more comprehensive distribution range.

From the graphs related to betweenness Centrality, we can see that the nodes with higher betweenness centrality tend to have higher speeds. The graphs show that the unblocked area has the widest betweenness Centrality distribution among the four regions. According to the literature [9], it is known that "usually the more accessible roads are generally far away from the city center," which means the further away from the city center the nodes are, the higher its betweenness Centrality. This also fits our intuition; the road network in suburban areas is not dense, so road points tend to act as the only "bridge" between neighboring road sections.

c. Closeness Centrality

We continue the same analysis steps similar to the previous two centralities for Closeness Centrality, but we get an unexpected finding here.

Closeness Centrality reflects the proximity of a node to other nodes. If a waypoint is relatively close to all additional waypoints, then the traffic speed of this waypoint is not too constrained by the different waypoints. It also captures the accessibility of a place at the city scale [10]. According to the definition of Newman (2001), the formula for Closeness centrality is:

$$C(x) = \frac{1}{\sum_y d(y,x)}$$

First, we plot the bar graph of closeness centrality to observe the distribution, as shown in the figure below.

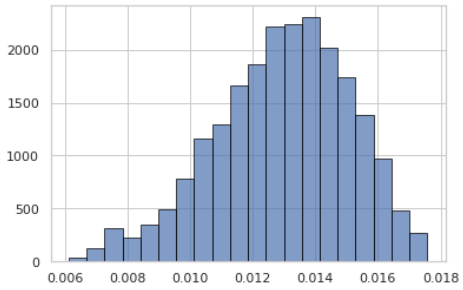

Figure 27. Closeness Centrality Distribution

From this figure, the overall Closeness centrality distribution is right-skewed. And the peaks are concentrated in the range of 0.013- 0.015. The distribution of closeness centrality is no longer like betweenness centrality and degree centrality but shows a trend of normal distribution. Next, we plot the closeness centrality as a categorical scatter plot. The result is shown in the figure below.

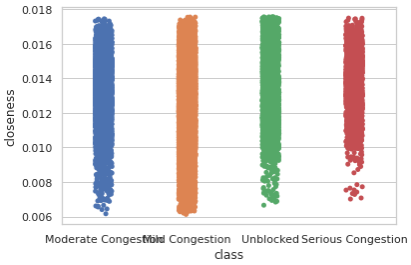

Figure 28. Closeness Centrality Categorical Scatter

This figure shows that as the congestion increases, the lower bound of the concentration range in different classes becomes higher and higher. The only exception is the unblocked region, which still has a high lower bound on the concentration range despite the low congestion level.

After that, we plot the density of different categories of closeness centrality. This is shown in the figure below.

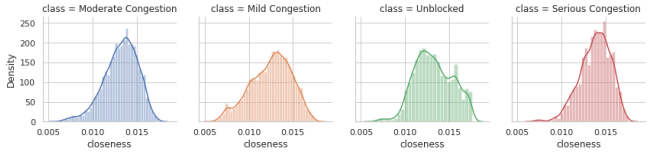

Figure 29. Closeness Centrality Distribution

From this figure, the right-skewed distribution in different classes becomes more and more potent as the congestion level increases.

The last is to draw the box diagram of the closeness centrality. This is shown in the figure below.

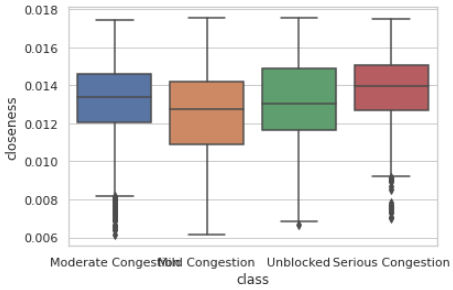

Figure 30. Closeness Centrality Boxplot

From this figure, the average closeness centrality of the remaining three classes, except the unblocked class, increases with the level of congestion. We believe that the anomaly in the degree centrality of unblocked is that we remove more nodes located in the "suburbs" in the preprocessing stage due to missing values.

From the above graphs, we can see that nodes with higher closeness centrality are more likely to be classified into Serious congestion. Since high-value areas of closeness centrality tend to be concentrated at the center of mass of the network [11], nodes with high closeness centrality in the road network tend to be closer to urban areas and more congested. This is consistent with our findings.

d. Eigenvector Centrality

Finally, we analyze the Eigenvector Centrality.

Eigenvector centrality considers that the importance of a node depends both on the number of its token nodes and on the importance of its neighboring nodes. The math formula is here:

$$X_v = \frac{1}{\lambda}\sum_{t\in M(v)} x_t = \frac{1}{\lambda}\sum_{t\in G} a_{v,t}x_t$$

As with the previous three centralities, we first plot the histogram of the Eigenvector Centrality, as shown in the figure below.

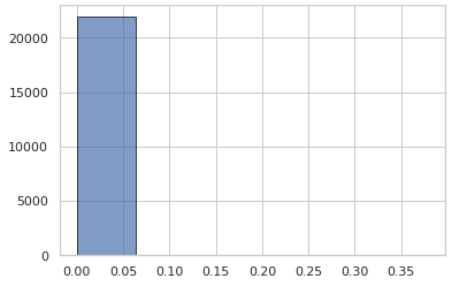

Figure 31. Eigenvector Centrality Distribution

This figure shows that the Eigenvector centrality of almost all the waypoints is concentrated in the range of 0-0.05

Then it is to plot the categorical scatter plot of Eigenvector Centrality.

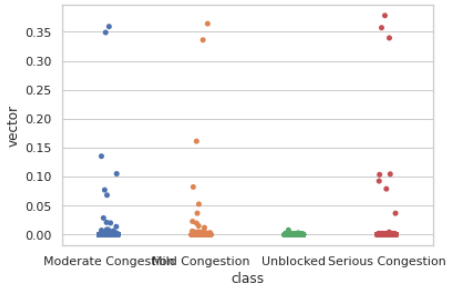

Figure 32. Eigenvector Centrality Categorical Scatter

This figure shows that the points of Eigenvector centrality in all four congestion classes are concentrated to 0. And the eigenvector centrality of a node in the unblocked class is more concentrated than others, which means all the nodes have an Eigenvector centrality around 0.

This is followed by a density map based on the four congestion categories

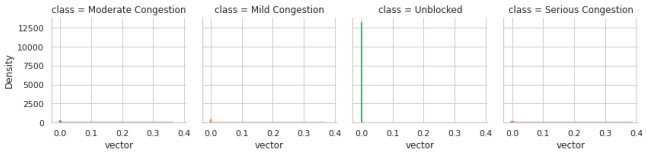

Figure 33. Eigenvector Centrality Distribution

Except for the unblocked class, we can see that the other three types are distributed in the interval of 0-0.4. And, since the distribution of unblocked types is relatively concentrated, only it has more significant values around 0.

The last is to plot the box diagram of Eigenvector Centrality, as shown in the figure below.

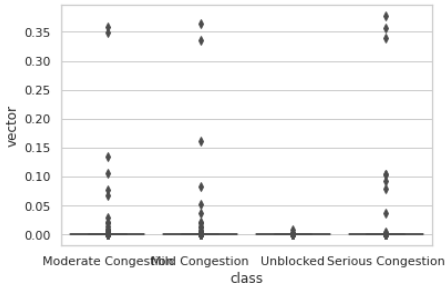

Figure 34. Eigenvector Centrality Boxplot

From this figure, the mean values of the eigenvector centrality for all four classifications are close to zero. And all classes except unblocked classes have particularly significant outliers.

Eigenvector centrality also describes the importance of a node. Unlike degree centrality, it considers the degree of neighboring nodes. However, according to the degree centrality analysis above, the contribution of a single node to the whole network is small due to the many nodes in the graph and the low degree of each node. Therefore, even considering the neighboring nodes, the eigenvector centrality can hardly describe the effect of individual nodes on the speed in our network, so we believe that there is no relationship between eigenvector centrality and rate. As for the generation of outliers, we speculate that very few nodes have multiple (>4) nodes connected to them, and when this node is considered, outliers are easily generated. According to the analysis in degree centrality, congestion tends to occur in nodes with a higher degree. This explains why there are no outliers in the unblocked class.

At the end of the Graph section of the analysis, we put four different centralities and speeds together to plot a heat map to explore a correlation between these four centralities and the rate.

The graph shows that speed and betweenness are positively correlated with a correlation coefficient of 0.24 since betweenness reflects whether a node is a "bridge" or not. This is consistent with our view that "bridges" tend to be suburban roads, and "bridges" tend to be "open." Speed is negatively correlated with the degree, with a correlation coefficient of -0.18. The higher the degree of a node, the more likely it is to be a "junction." Speed is negatively correlated with closeness, with a correlation coefficient of -0.058. The higher the closeness of a node, the more likely it is to be a "junction." The higher the closeness of a node, the closer it is to an urban area, and the more likely it is to be congested. Therefore, the higher the closeness, the lower the traffic speed. The correlation coefficient of the vector is approximately equal to 0. This is consistent with our view that vector is not related to traffic speed.

V. MODEL

The model we use in this project is GAT.

A. *Introduction*

The core idea of GAT, i.e., Graph attention networks, is to introduce an attention mechanism into the graph convolutional network. Since the input of our project is graph-based traffic speed data, which is often a sequence of temporal data, the general graph convolutional network (GCN) cannot make effective extraction of temporal information from the nodes. In this regard, we use a GAT network consisting of a Graph Attentional Layer, which aggregates the information of neighbor nodes in spatial latitude, and later aggregates the information of neighbor nodes in temporal latitude through a Multi-Head-Attention mechanism, to train the model and make predictions.

B. *Model Input*

Traffic data processing is not quite the same as standard data processing. We use the Sliding Window approach since our traffic data is a series of time-series data. The following are the specific processing steps.

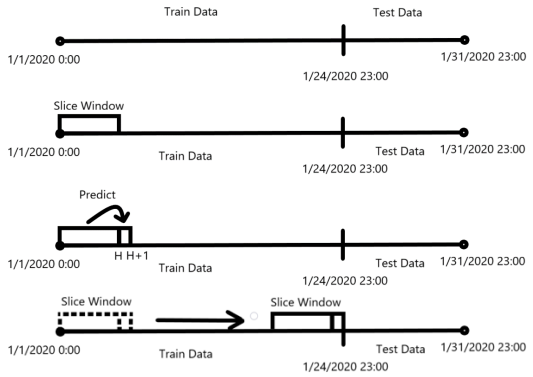

Figure 35. Sliding Window on data

- The data for each hour from January 1, 2020 at 00:00 to January 31, 2020 at 23:00 (744 hours in total) were segmented by using the first 24 days (576 hours) as the training set and the last 7 days (168 hours) as the test set.
- Artificially define a Sliding Window size, in this project, the window size is 24 hours
- Using the data within the window (called historical data, H), predict the traffic flow at the moment H+1
- Repeat step 3 and move this window by one time unit at a time until all training samples are obtained

C. *Background*

1. Attention Mechanism

From the nomenclature of the Attention Mechanism, it is clear that it is originated from the Human Visual Attention Mechanism. The Human Visual Attention Mechanism is a signal processing mechanism in the brain critical to human vision. The human eye quickly scans the global image to obtain the target area that needs to be focused on, and then devotes more attention to this area to get more detailed information about the target that needs to be focused on, while suppressing other useless information. The Attention Mechanism in deep learning is essentially similar to the human visual attention mechanism. The core goal is to select the more critical information to the current task goal from a large amount of information.

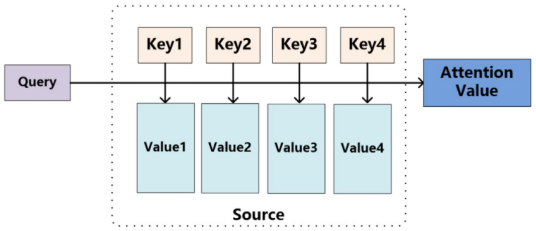

Figure 36. Attention Mechanism

We can look at the Attention mechanism in this way (as shown above): Imagine the constituent elements in Source as a series of <*Key, Value*> data pairs. At this time, given the Query of an element in Target, the weight coefficient of *Value* corresponding to each *Key* is obtained by calculating the similarity between the *Query* and each *Key*. Then the *Value* is weighted and summed to obtain the Attention value. So essentially, the Attention mechanism is a weighted summation of the *Value* of the elements in the Source, while the Query and *Key* are used to calculate the weight coefficients of the corresponding *Value*. The formula is

$$Attention(\text{ Query, Source }) = \sum_{i=1}^{L_x} Similarity(Query, \text{Key }_i) * \text{ Value }_i$$

where $L_x = |Source|$, that is L_x represents the length of the source. In this project, the set of Source is the set of neighboring points of the traffic road network, and the key is the same as the value, which is the speed of each node at a specific time.

As for the specific calculation process of the Attention Mechanism in this project, we can summarize it as follows:

1. We calculate the similarity between the two in the set of neighboring points based on the speed value.

$$\text{Similarity}(\text{ Query, Key }_i) = \text{ Query } \cdot \text{ Key }_i$$

2. normalize the raw scores in step 1 to derive the weighting coefficients

$$a_i = Softmax(Sim_i) = \frac{e^{Sim_i}}{\sum_{j=1}^{L_x} e^{Sim_j}}$$

3. Weighted summation of speed value according to weighting coefficients, resulting in attention value

All steps are shown in the figure below:

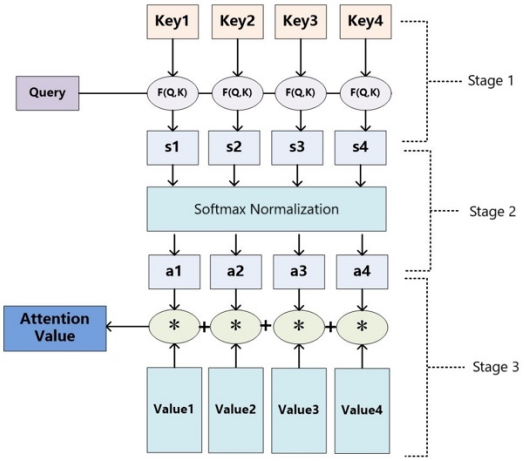

Figure 37. Weight Summation

2. Multi-Headed-Attention mechanism

It is not enough to use the Attention Mechanism mentioned above to handle the information between nodes. This is because the attention mechanism only considers the spatial connectivity information of nodes, while ignoring the temporal connectivity information of nodes. Also, according to the above introduction, we can find that if we use the attention mechanism alone, the model will focus too much on its own position when encoding the current situation. Therefore, we use the Multi-Head-Attention Mechanism, which allows the model to process the data temporally and consider the information of neighboring nodes. The Multi-Head-Attention structure is shown in the following figure.

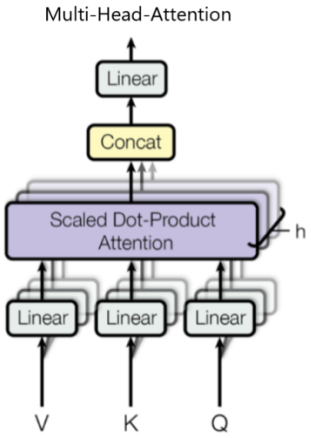

Figure 38. Multi-Headed-Attention mechanism

We can see that the Multi-Head-Attention mechanism is actually a multi-group attention process for the original input sequence; then, the attention value of each group is stitched together for a linear transformation to obtain the final output. The calculation formula is:

$$MultiHead(Q,K,V) = Concat(head_1, \ldots, head_h)W^O$$

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

, where $W_i^Q \in R^{d_{\text{mdl}} \times d_k}$, $W_i^K \in R^{d_{\text{mdl}} \times d_k}$, $W_i^V \in R^{d_{\text{mdl}} \times d_v}$, $W^O \in R^{hd_v \times d_{\text{mdl}}}$

3. Graph Attentional Layer

The Graph Attentional Layer is the basis of the whole GAT model. The input of a single Graph Attentional Layer is a set of temporal vectors of nodes.

$$h = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_N}\}, \overrightarrow{h_i} \in R^F$$

, where N denotes, the number of neighboring nodes of the node and F denotes the corresponding latitude of the timing vector.

The output of each layer is a new set of feature vectors for the nodes:

$$h' = \{\overrightarrow{h_1'}, \overrightarrow{h_2'}, \ldots, \overrightarrow{h_N'}\}, \overrightarrow{h_i'} \in R^{F'}$$

, where F' denotes the new feature vector latitude of the node (not equal to F).

The structure of a graph attentional layer is shown in the following Figure.

Specifically, the Graph Attentional Layer first performs attention processing based on the input set of temporal vectors of neighboring nodes to calculate the correlation degree between nodes. It then uses the multi-head-attention mechanism to perform differentiated information aggregation on neighboring nodes. The specific steps are as follows.

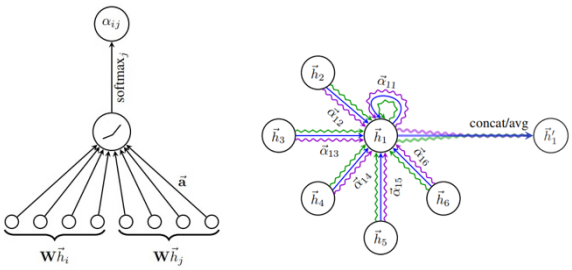

Figure 39. Graph attentional layer

- Model Input
- Linear Transformation

To obtain sufficient expressiveness, the input features are transformed into higher-level features. We use a learnable linear transformation $\boldsymbol{W} \in R^{F' \times F}$, to map the model input to a higher dimensional space

- Attention Mechanism

After using the linear transformation, we employ an attention function to obtain the attention distribution.

$$e_{ij} = a\left(W\overrightarrow{h_i}, W\overrightarrow{h_j}\right)$$

, where e_{ij} stand for the importance of features for node j to node i. The above equation is a general case where the graph's structure is not considered, and each node

participates in the attention computation of other nodes. In fact, in this project, we use masked attention (using the adjacency matrix as a mask) to introduce information about the structure of the graph-we compute e_{ij} only for node j in the set N_i of neighboring nodes belonging to node i.

The attention scoring function we use is specified by splicing the high-dimensional spatial representations of node i and node j, followed by a linear transformation $\vec{a} \in R^{(2F')}$, and finally a nonlinear activation function LeakyReLU:

$$e_{ij} = LeakyReLU\left(\vec{a}^T\left[W\overrightarrow{h_i}|W\overrightarrow{h_j}\right]\right)$$

- Softmax Normalization

To make the weight coefficients on different nodes easy to compare, we normalize the weights of each node to obtain the attention distribution.

$$\alpha_{ij} = Softmax_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in N_i}\exp(e_{ik})}$$

- Model output

The obtained attention distribution is used to compute a weighted average of the feature representations of the neighboring nodes of a node, and the final output is obtained by stitching each sub-attention structure using a Multi-Headed-Attention Mechanism.

D. *Our Model*

Our model diagram is as follows.

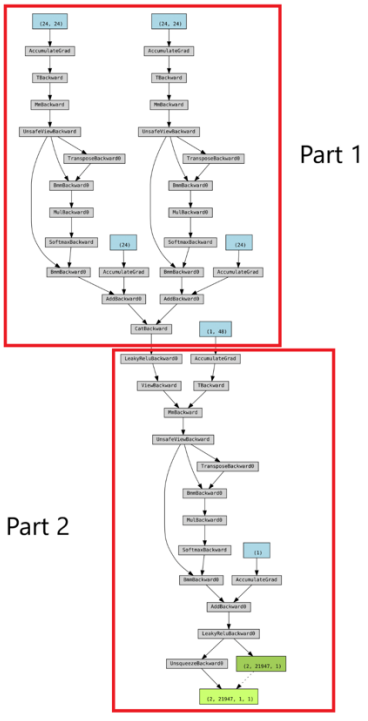

Figure 40. Model Structure

As seen from the above figure, our whole GAT model consists of two parts. In the first part, the attention value of each node is calculated using Graph Attentional Layer. And in the second part, the attention value of each node is aggregated and output using Graph Attentional Layer according to the Multi-Head-Attention Mechanism.

VI. Model Comparison and Analysis

We used the Prophet statistical model and Random Forest Model to compare and evaluate the GAT model. The evaluation parameters are MAE, MAPE, and RMSE. It can be seen from the table below that our GAT model performs well.

TABLE II. Model Evaluation Matric

	MAE	*MAPE*	*RMSE*
GAT	7.1	0.29	9.55
Prophet	142.7	452.4	215.0
Random Forest	14.4	35.9	93.4

A. *Prophet*

Prophet is published by Facebook's Core Data Science team. It depends on a contribution model where non-linear trends are fit with weekly and yearly seasonality and plus holidays. PROPHET is strong to missing data, capturing the shifts in the trend and large outliers. In addition, it gets a reasonable estimate of the mixed data without spending manual effort [12].

B. *Random Forest*

Random forest is an algorithm that integrates multiple trees through blended learning. Its basic unit is a decision tree, and its essence belongs to Ensemble Learning, a significant branch of machine learning. Each decision tree is a classifier (assuming we are talking about a classification problem), so N trees will have N classification results for one input sample. The random forest integrates all the classification votes and designates the category with the most votes as the final output, one of the most straightforward Bagging ideas. [13]

C. *Evaluation Matric*

1. MAE

In statistics, mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus\ X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as [13]:

$$MAE = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n} = \frac{\sum_{i=1}^{n}|e_i|}{n}$$

2. MAPE

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics. It usually expresses the accuracy as a ratio defined by the formula:

$$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

, where At is the actual value and Ft is the forecast value. Their difference is divided by the actual value At. The absolute value in this ratio is summed for every forecasted point in time and divided by the number of fitted points n. [15]

3. RMSE

Root mean square error (RMSE) is a standard measure of the difference between the value predicted by a model or estimator (sample or overall value) and the observed value. RMSE represents the square root of the second sample moment of the difference between the predicted and observed values or the quadratic mean of these differences. When computed on a sample of data used for estimation, these deviations are called residuals and are referred to as errors (or prediction errors) when calculated out-of-sample. RMSE is used to aggregate the magnitude of prediction errors for various data points into a single measure of predictive power. RMSE is a measure of accuracy used to compare prediction errors across models for a given data set rather than between data sets, because it is related to size correlation. [16]

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

4. Real World Performance

The following figure shows the predicted value versus the actual value for a particular node timing. The red color indicates the expected value, and the blue is the actual value. The horizontal coordinates indicate the time series of the test set (every 24 hours for the last 7 days), and the vertical coordinates are the velocity values

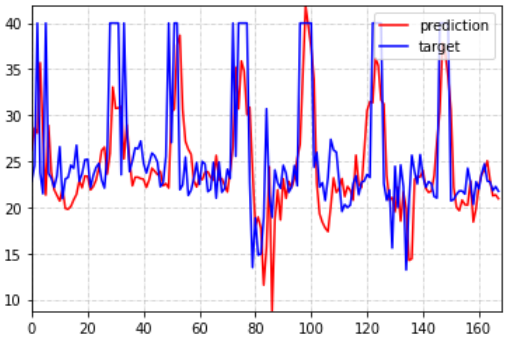

Figure 41. Real Word Performance

As we can see from the above figure, the model can learn the temporal trend of the data relatively well, and the predicted speed is close to the actual value. The model performs well on the test set.

D. ***Model Analysis***

1. Advantages

We believe that the GAT model has several advantages.

GAT is efficient. GAT does not need to use complex matrix operations such as eigenvalue decomposition compared with other graph models. The time complexity of the Graph Attentional Layer is $O\left(|V|FF' + |E|F'\right)$, which is the same as that of GCN. In the expression, |V| and |E| denote the number of points and the number of edges in the graph, respectively. Since our dataset is relatively large (a total of 21947 nodes, 74783 edges, and 744 data on each node), therefore, the time spent on training would be astronomical if other graph models were used.

GAT has a stronger representation capability. Compared with GCN, the importance of each node in GAT can be different, which fits well with one of the characteristics of our dataset: different sub-maps contribute differently to the whole map (different speeds of vehicles in different sections). At the same time, GAT can extract information about the data in terms of time latitude, which is also in line with the characteristics of traffic flow data - a sequence of temporal data.

The GAT model considers all neighboring nodes and does not assume the internal order of these nodes. All nodes of the domain in GAT do not need to be ordered and share the convolution kernel parameters.

2. Limatition

At the same time, we believe that the GAT has several disadvantages.

Unlike GNN, GraphSAGE, GAT directly selects first-order adjacent nodes as domain nodes. This is not very similar to reality; when traffic congestion occurs, it often impacts the first-order neighbor nodes, and the second-order, third-order are ignored.

The GAT calculation time is still too long. We are using an A40 graphics card, but it still takes nearly 20 minutes to run an epoch, making it difficult to adjust the optimal parameters.

VII. CONCLUSION

Through this project, we understood and explored the task of analyzing and predicting traffic flow data, understood and applied the knowledge of graph neural networks, built proper GAT neural networks to cope with the task of traffic speed prediction in New York City, and the model has good performance. However, our model still has some limitations that need to be further overcome in the subsequent research work.

ACKNOWLEDGMENT

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to thank my professor, Professor Bhupesh Shetty , whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would particularly like to acknowledge my group mate for their wonderful collaboration and patient support

Finally, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me.

REFERENCES

[1] Goode, L. (2011, June 17). *Worth it? an app to get a cab*. The Wall Street Journal, www.wsj.com/articles/BL-DGB-22634

[2] Uber "Uber Movement: Let's Find Smarter Ways Forward, Together." *Uber Movement*, movement.uber.com/?lang=en-US..

[3] Gilbertson, Jordan. "Introducing Uber Movement." *Uber Newsroom*, 8 Jan. 2017, www.uber.com/newsroom/introducing-uber-movement-2/.

[4] Anderson, Mark. "Global Positioning Tech Inspires Do-It-Yourself Mapping Project." *National Geographic News*. October 2006-10-18, www.nationalgeographic.com/news/2006/10/061018-street-maps.html.

[5] Wiki Contributors. "Map Features." *Map Features - OpenStreetMap Wiki*, https://wiki.openstreetmap.org/wiki/Map_features.

[6] Pearson, Mackenzie, Javier Sagastuy, and Sofia Samaniego. "Traffic flow analysis using uber movement data." htts://snap. stanford. edu/class/projects (2017).

[7] TLC Trip Record Data - TLC. (n.d.). TLC Trip Record Data. Retrieved January 31, 2022, www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[8] Abaya, Ernesto B., et al. Instantaneous Fuel Consumption Models of Light-Duty Vehicles and a Case Study on the Fuel Consumption at Different Traffic Conditions in Metro Manila Using Shepard's Interpolation Method. No. 2018-01-0075. SAE Technical Paper, 2018.

[9] Tomko M, Winter S, Claramunt C. Experiential Hieranrchies of Streets [J]. Computers Environment and Urban Systems, 2008, 32(1): 41-52.

[10] Jayaweera, I. M. L. N., Perera, K. K. K. R., & Munasinghe, J. (2017). Centrality measures to identify traffic congestion on road networks: A case study of Sri Lanka. IOSR Journal of Mathematics (IOSRJM).

[11] Crucitti P, Latora V, Porta S. Centrality in networks of urban streets[J]. Chaos: An Interdisciplinary Journal of Nonlinear Science, 2006, 16(1): 015113 DOI:10.1063/1.2150162.

[12] Taylor, Sean J., and Benjamin Letham. "Forecasting at scale." The American Statistician 72.1 (2018): 37-45

[13] Tavish. "Random Forest Algorithm | Introduction To Random Forest". Analytics Vidhya, 25 juni 2020, www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified.

[14] [14] Willmott, Cort J., and Kenji Matsuura. "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance." Climate research 30.1 (2005): 79-82

[15] De Myttenaere, Arnaud, et al. "Mean absolute percentage error for regression models." Neurocomputing 192 (2016): 38-48

[16] Hyndman, Rob J., and Anne B. Koehler. "Another look at measures of forecast accuracy." International journal of forecasting 22.4 (2006): 679-688