Taxi Demand Prediction From Previous Trip Records and Weather Information

Tahjid Ashfaque Mostafa[1][0000−0003−3033−635X]

University of Alberta Department of Computing Science, Edmonton AB, Canada tahjid@ualberta.ca
https://www.ualberta.ca/computing-science

Abstract. In this paper, we compare the performances of different machine learning and deep learning models in predicting taxi demand for a particular time slot in a particular location. We used yellow and green cab trip records from NYC Taxi dataset for our experiments. K-Means clustering was used on pick up locations to find hotspots after preprocessing and cleaning the data. Various weather features for location were also added to our data as features.

1 Introduction

Taxi cabs provide an important mode of transportation for ordinary commuters in their day to day lives. A large number of people all over the world are dependent on taxi cabs for their livelihood. We propose a method to help taxi drivers and companies predict taxi demand in particular areas beforehand, thus enabling them to be nearer to the areas with most demand, helping organize the taxi fleet and minimizing the wait time for both passengers and drivers. It can be vital in improving user experience and strengthening business performance, specially since the demand for taxi cabs have been steadily declining since the advent of ride sharing services. This might play a vital role in ensuring the sustainability of the taxi companies.

There have been some work in this area before. Moreira et al.[1] predicted the the taxi passenger demand for a 30 minute horizon using data from 441 vehicles from the city of Porto, Portugal. Zhao et al[2] used three separate predictors on yellow cab and for hire vehicle data taken from NYC taxi dataset after measuring the demand uncertainty at a building block level. They claim that their approach had the best accuracy of 89% when using Markov predictors. Xu et al.[3] proposed a sequence learning method using LSTM Recurrent Neural Networks predict taxi demand in real time. They divide the city into small blocks and added relevant information like weather and time. Wong et al.[4] propose a two level model for taxi movement in congested road networks. Yang et al.[5] propose a simultaneous equation system of passenger demand, taxi utilization and level of services based on a taxi service situation found in the urban area of Hong Kong over the ten years

2 Problem Statement and Challenges Faced

Our ultimate goal is to predict the number of taxi pickups for a specific region at a specific time period given history of pickups in that area and the weather information. It is basically a time series forecasting problem, given the information about moment $t-1$, we have to predict about moment t. We need our solution to be able to predict in real time. If the computation takes too long the results will become irrelevant. So the latency should be moderate to low. One of the major problems we faced we faced was the size of the dataset used. Loading data with standard python libraries like *Pandas* will cause memory issues. To get around this issue, we used *dask* library. Dask is very convenient when dealing with data whose size is larger than ram size, it loads data to ram in blocks. Only the necessary data are loaded and as soon as it is preocessed, dask empties the ram.

Another problem we faced was identifying relevant features to use. Choosing more features than needed might negatively impact the prediction performance.

3 Methodology and Experimental Evaluation

3.1 Data

Trip Data For trip data, we used New York City (NYC) taxi trips dataset publicly released by the Taxi and Limousine Commission (TLC)[7]. This dataset contains trip information about taxicabs and for hire vehicles like Uber or Lyft rides happening in NYC. There are mainly two types of taxi cabs in NYC, Yellow (medallion) taxis, which are able to pick up passengers anywhere in the five boroughs and Green(Street hail livery vehicles) taxis, commonly known as "boro taxi", which can pick passengers in Upper Manhattan, the Bronx, Brooklyn, Queens (excluding LaGuardia Airport and John F. Kennedy International Airport), and Staten Island. Both types have similar fare structure. We used yellow and green taxi data in our analysis. Since this dataset is very large, it was not computationally feasible for us to use the whole dataset. We decided to use only a subset of the dataset. We only considered trips happening in the month of January 2017, which included 9,710,124 yellow cab trips and 1,070,261 green cab trips. For each trip, we considered pickup and drop off date and time(the exact moment the meter was engaged or disengaged), pickup and drop off location Id, passenger count entered by the driver, trip distance and fare amount. We discarded the rest of the data like tip amount, tolls amount, payment type etc.

Location Data Previously each trip record also contained the exact latitude and longitude of the pick up and drop off. But due to security and privacy concerns, from June 2016 onward, these data were removed. Instead, the whole city was divided into 260 zones, each having a unique location Id, and the Id for pick up and drop off locations were provided with each trip. The zones are not of similar size or shape. Figure 1 shows the NYC taxi zones. To get around

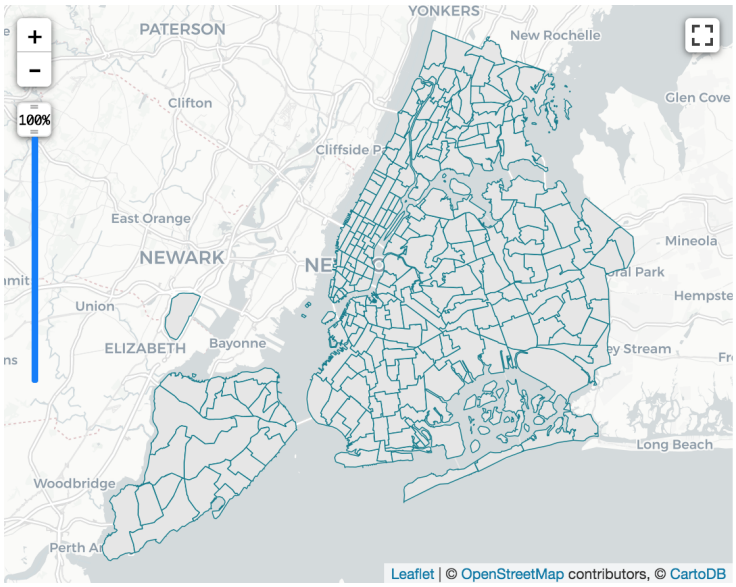


Fig. 1. NYC Taxi Zones

this issue, we acquired the latitude and longitudes of each location from the boundaries of each zone taken from [8] using Python Shapefile Library(pyshp). This data also contained additional information like shape of the zone , borough the zone belongs to etc, but we only considered the latitude and longitude.

Weather Data We collected the weather information for the month of January, 2017 from [9]. The data contains extreme maximum, maximum,extreme minimum, minimum and average temperature for the month, total sunshine for the period, snowfall, precipitation, heating degree days(Days where the mean of the temperature is below 65 degree Fahrenheit), cooling degree days(Days where the mean of the temperature is above 65 degree Fahrenheit), extreme maximum snowfall, precipitation and snow depth, latitude and longitude for the location of the data.

3.2 Data preprocessing and outlier removal

The taxi data has a lot of outliers and erroneous points. We cleaned the data by removing outliers based on the steps proposed by [11, 12] and also added some extra columns. We performed our preprocessing on both yellow and green cab data. But to avoid redundancy and satisfy space constraints, graphs used in this section are from yellow cab data unless stated otherwise.

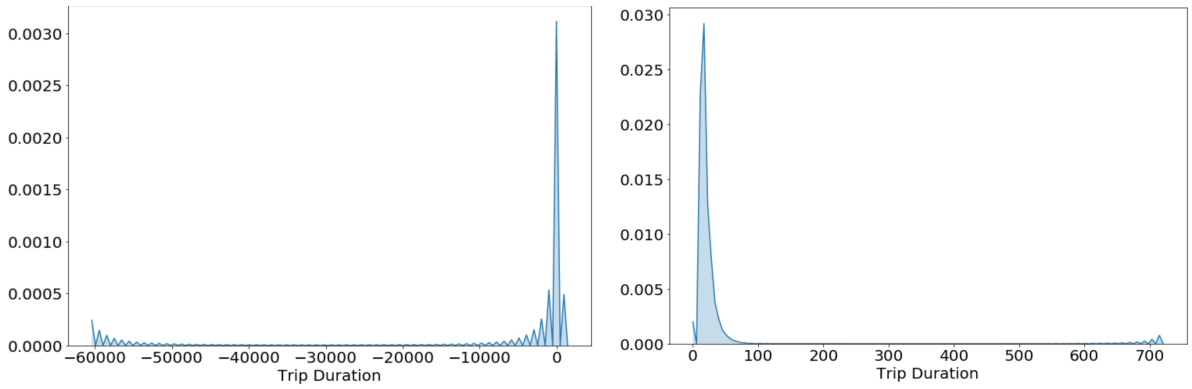
Adding Trip Duration and Speed We first added two new values to our data, trip duration and speed. Trip duration is calculated by the difference between pick up and drop off datetime. The times were converted to Unix format for ease of computation, which is basically a system for identifying a particular point in time. It is the number of seconds that have elapsed since the Unix epoch, that is the time 00:00:00 UTC on 1 January 1970, minus leap seconds [10]. The formula for trip duration is given in Equation 1

$$Trip\ Duration(Minutes) = \frac{Pick\ Up\ Time\ -\ Drop\ Off\ Time}{60} \tag{1}$$

Then we calculated the speed of the trip in miles/hour from trip distance and trip duration using Equation 2

$$Speed(Miles/Hour) = \frac{Trip\ Distance}{Trip\ Duration} \times 60 \tag{2}$$

Trip Duration Outlier Removal Figure 2(a) shows a KDE distribution plot for our current trip duration for yellow cab data. As we can see the distribution is erroneous. There are negative values, which means there are data in our table with drop off times before pick up times. We



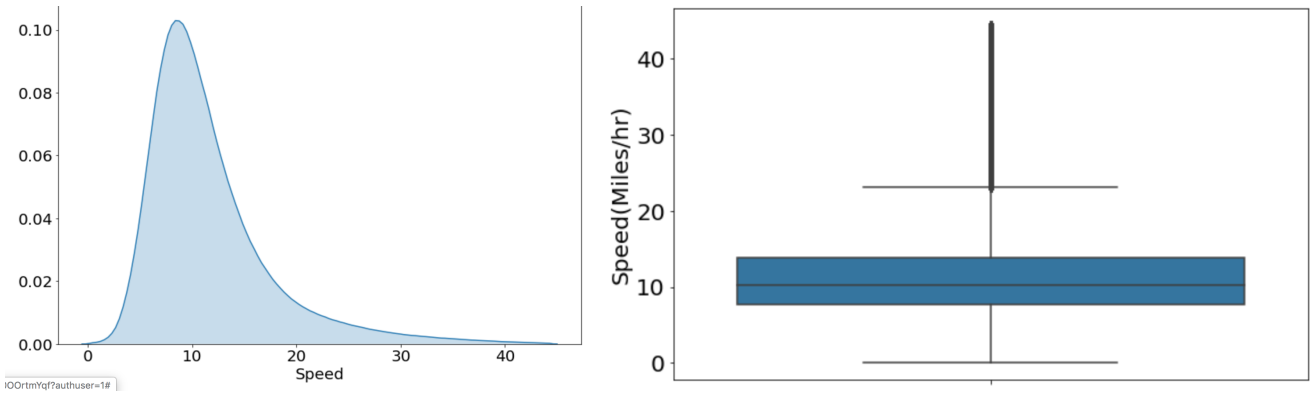
(a) Before Filtering (b) After Filtering

Fig. 2. KDE Distribution plots for trip duration of yellow taxi data

calculated a quantile distribution for our trip duration, and found that the data ranged from

-60400.25 to 1439.9 minutes for yellow cabs and 0 to 1439.9 minutes for green cab data. But according NYC rules, the maximum allowed duration of a trip is 12 hours which is 720 minutes. So we removed all trips with negative values and values larger than 720 for trip duration. We can see from our quantile distribution that only 10% of the trips are above 25 minutes for both yellow and green cabs.

Speed Outlier Removal Figure 3(a) shows a KDE distribution of speed for Yellow cabs. As we can see it's pretty standard, but there are some values which are negative. We also found from quantile distribution that there are some very high values in the 100th percentile. For yellow cabs it's 2131.94 miles/hour and for green cabs it's 1074.28 miles/hour, both of which are practically impossible. So we set a limit on the speed so that the highest acceptable values are the 99.9th percentile value, which is 44.42 miles/hour for yellow cabs and 42.37 miles/hour for green cabs. We discarded all trips with speed exceeding the limit and trips with a negative speed. Figure 3(b) shows a box plot of speed after the filtering. We note that the average speed is 11.71 miles/hour or 1.95 miles/10 minutes for yellow cabs and 12.45 miles/hour or 2.07 miles/10 minutes for green cabs.



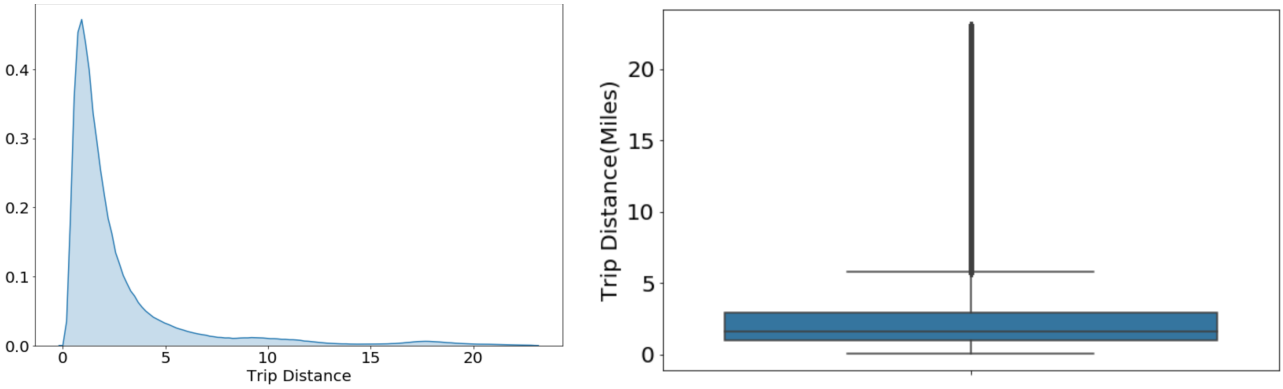
(a) KDE Before Filtering

(b) BoxPlot After Filtering

Fig. 3. Speed of yellow taxi data

Trip Distance Outlier Removal We analysed the trip distance for the data and found that there were some unrealistic values, some values are negative, some values are too large. Figure 4(a) shows the distribution of trip distance before filtering. For yellow cabs, the 100th percentile value is 122.25 miles and for green cabs it's 81.7 miles. Since both these values seem unrealistic and occur very sparsely, we considered theese as outliers and set an upper limit with the 99.9th

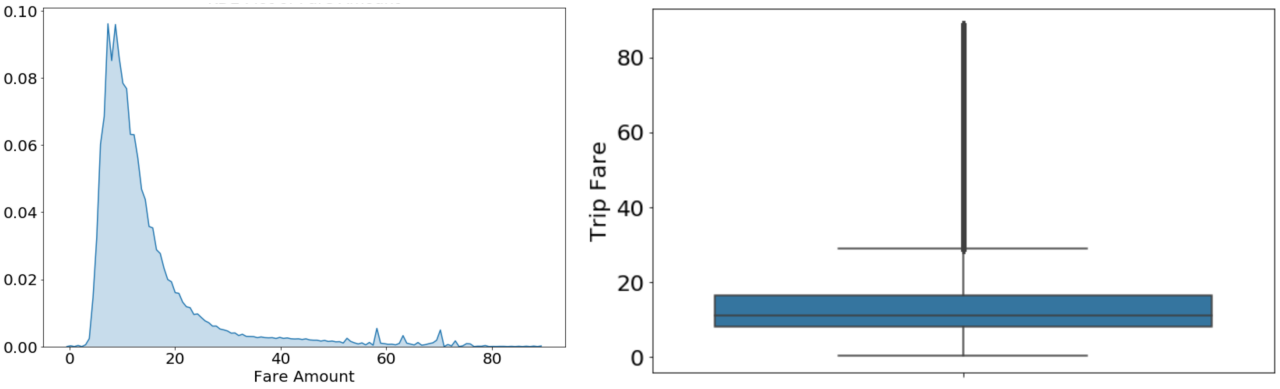
percentile values, which is 22.9 for both yellow and green cabs. Figure 4(b) shows most trip distances are within 5 miles.



(a) KDE Before Filtering (b) BoxPlot After Filtering

Fig. 4. Trip distance of yellow taxi data

Fare Amount Outlier Removal We performed similar analysis of fare amount and found that the 100th percentile fare is 3009.8 USD for yellow cabs and 325.8 USD for green cabs. We set the upper limit to 99.9 percentile values which is 88.56 USD for yellow cabs and 72.9 for green cabs. We removed all trips with negative fare amount and fare amount higher than the upper limit. Figure 5 shows us distribution of trip fares before and after filtering.



(a) KDE Before Filtering (b) BoxPlot After Filtering

Fig. 5. Trip Fare of yellow taxi data

Passenger Count Outlier Removal The passenger counts for the trips are often reported by drivers themselves, so it is not very accurate. Figure 6 shows the distribution of passenger

count for yellow cabs. We can see that some trips have zero passengers which does not make sense. We only consider trips with positive passenger count and discard the rest.

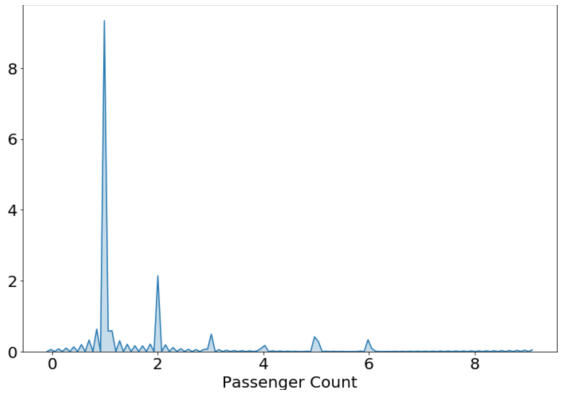


Fig. 6. Passenger Count Distribution

Ensuring all pick ups and drop offs are inside NYC There are cases where the pick up or the drop off location of a trip is outside the city bounds. We would like to consider only the trips inside the city. We can use Equation 3 to determine whether a point is inside NYC boundaries or not by comparing the latitude and longitude of the point with NYC city boundaries.

$$\begin{aligned} 40.5774 <= Latitude <= 40.9176 \\ -74.15 <= Longitude <= -73.7004 \end{aligned} \tag{3}$$

Figure 7 shows some pick up points outside NYC, the green pointers represent green cabs and yellow pointers represent yellow cabs. We removed all trips with pick up or drop off points outside NYC.

After the cleaning process, we had removed 119,610 erroneous trip records for yellow cabs and 27162 trip records for green cabs. The remaining records are 98% of the original dataset for yellow cabs and 97% for green cabs.

3.3 Clustering

We clustered the pick up locations for identifying possible pick up hotspot locations. We did not use the taxi zones as location identifier because pickup hotspots might span across multiple zones, thus the zones might not accurately represent pickup density. But clustering will be more accurate because they will be limited by the zone boundary. For clustering we use a variant of **K-Means Clustering** known as **MiniBatchKMeans**, which clusters small random

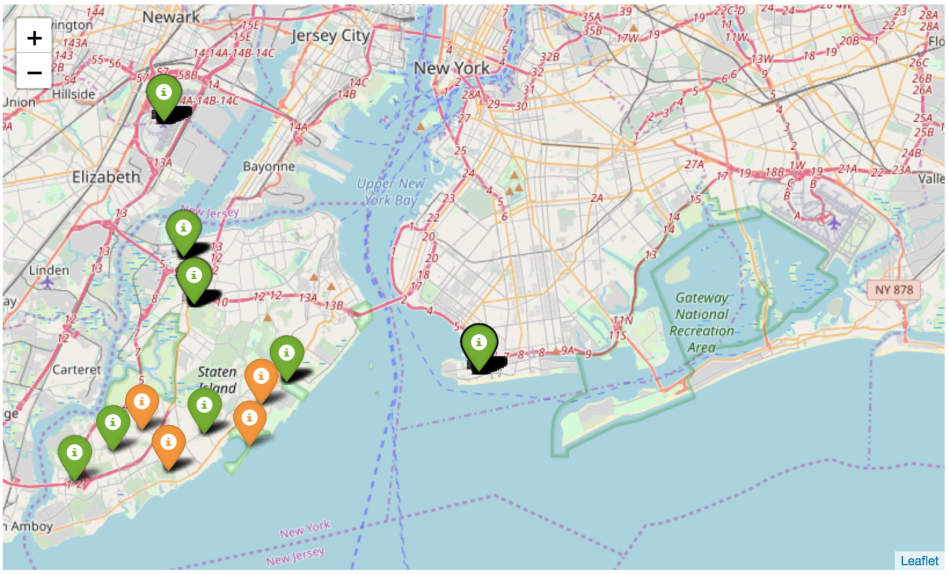


Fig. 7. Pickup points outside NYC for Yellow and Green Cabs

samples of data to cluster at a time until the clusters converge. Since the dataset is so large and we lacked the computation resources to process the whole data simultaneously, we choose MiniBatchKMeans with a batch size of 10,000 instead of regular K-Means. We didn't use latest density based or hierarchical clustering techniques like DBSCAN or OPTICS because they are not suited to the problem. In some areas the density of pick ups is very high, it is not possible to accurately differentiate clusters.These algorithms will identify one or two very large clusters in the whole city, which is not helpful for our purpose.

One of the main issues with K-Means algorithm is the number of clusters K has to be predetermined. K-Means creates clusters of roughly the same size, size referring to the number of data points, not the area of the cluster. So in high pick up density areas like Manhattan, the area of the clusters will be smaller compared to outskirts of the city. We wanted our cluster size around 2 miles and inter-cluster distance to be around .5 mile.

For finding the best value of K, we created clusters with multiple values of K and computed the straight line distance between the [latitude, longitude] pairs of the centers of the clusters with the python library **gpxpy**. We achieved the best results for K = 30 for yellow cabs and K = 40 for green cabs. For these values of K, the minimum distance between clusters was closest to .5 miles. Figure 8 shows the centers for the detected clusters and their locations. Yellow points represent clusters for yellow cab trip data and green pointers represent clusters for green can trip data. After that each trip record was assigned to its nearest cluster. Figure 9 shows the resultant clusters. The red star symbols represent cluster centers. The colored circles around it

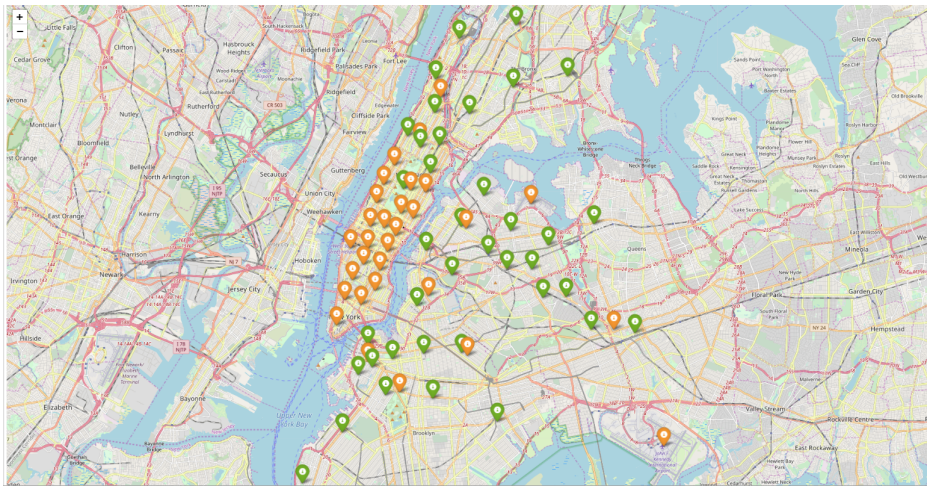
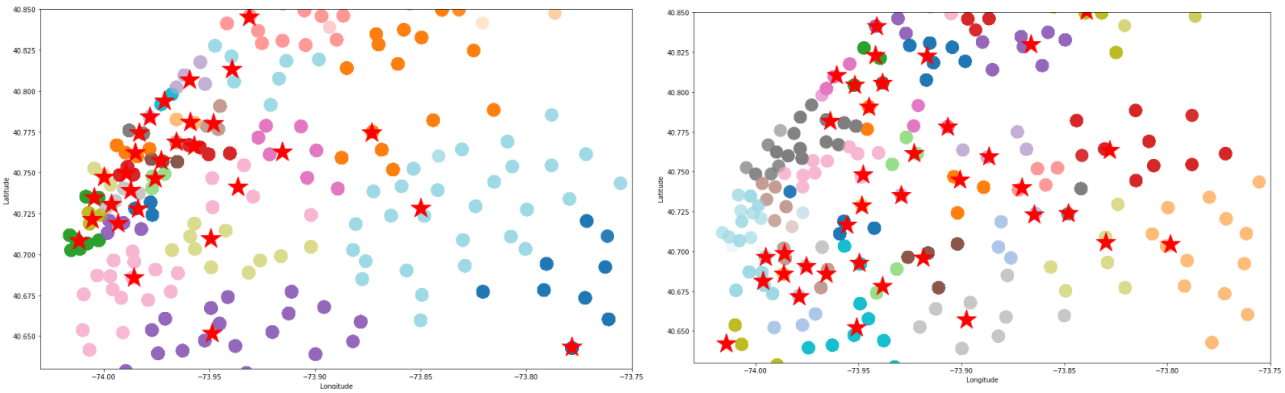


Fig. 8. Cluster Centers

represent clusters. Same colored points belong to the same clusters. The clusters are not more compact as we do not have the exact pick up locations, we only have the location of the taxi zones the pic ups occurred in. Thus many trips have the same pickup points. Each point in the graph represents one taxi zone. If we had the exact pickup points the clusters would have been more compact.



(a)Yellow cab trip data

(b) Green cab trip data

Fig. 9. Resultant Clusters

3.4 Counting the pickups per cluster per time period

Dividing time period into slots We divided the whole time period, which in our case in one month into 10 minute slots. For January 2017 data, each time slot is the time amount of seconds elapsed since midnight of the first day of the month. The total number of unique time slots for January is $\frac{24(Hours)*60(Minutes)}{10(Length\ of\ time\ slot\ in\ minutes)} \times 31(Number\ of\ days) = 4464$.

Then we assign each trip to a time slot based on Equation 4

$$Time\ Slot = \frac{Pick\ up\ time\ -\ Start\ of\ month}{60 * 10} \tag{4}$$

Counting trips per cluster per time slot Then we group our data by cluster and time slots and count the number of pickups, so we have the number of pickups happening for each cluster for each time slot. Figure 10 shows a sample of the resultant data structure for yellow cabs. *pickup_cluster* column represents the cluster Id, *time_bin* represents the time slots and the last column shows the number of pick ups for that time slot.

		Number of trips
pickup_cluster	time_bin	
0	0	63
	1	135
	2	158
	3	201
	4	180

Fig. 10. Number of trips per cluster per time slot

Filling missing values After dividing our trips into ten minutes slots, there might be situations where for a particular cluster there are no trips in a particular time slot. We can get around this issue in two ways.

1. Filling missing values with 0
2. Smoothing

If we assume the number of pickups in 4 consecutive time slots to be $\_, \_, \_, x$, i.e. the first three have no pickups and the last slot has x pickups, smoothing is distributing the pickups across all slots, like $x/4, x/4, x/4, x/4$ and filling with zero is just putting 0 in the empty slots, like $0, 0, 0, x$. But smoothing would disrupt the pattern of the data and is not suitable for our problem since we want to predict pickups and smoothing changing the values of the pickups might lead to wrong learning for our models. So we chose the first method and filled in all missing values with 0.

3.5 Feature engineering

Pickups happening in last 5 time slots The first feature we used is the number of pickups happening in a the last 5 time slots. So to predict the number of pickups happening in cluster

C in time slot t, we will take the number of pickups in C for time time slots $t-1, t-2, t-3, t-4\, and\, t-5$.

Weighted Moving Average The next feature is the weighted moving average of the past 5 pickups using Equation 5.

$$WMA = \frac{n \times P(t-1) + (n-1) \times P(t-2).....P(t-(n))}{\frac{n \times (n+1)}{2}} \tag{5}$$

We used weighted moving average because the nearest time slots are more likely to be similar to the time slot being considered. The nearest time slots are assigned more weight with the value of n and it keeps decreasing with each previous slot. n is the hyper parameter which is also used to control the window size, i.e. how many previous time slots we want to consider.

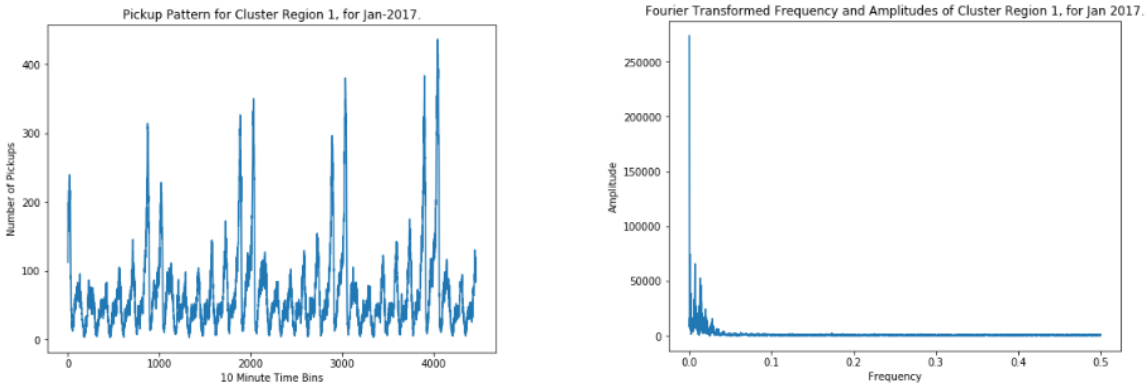
Cluster Center We also used latitude and longitude of the cluster center as features.

Time Slot We added the unique time slot number as a feature, since the number of pickups have a relation with the time.

Day of the Week We assigned a number to each day of the week, Sunday = 0, Monday = 1, Tuesday = 2, Wednesday = 3, Thursday = 4, Friday = 5, Saturday = 6. For a time slot, we calculated which day of the week it falls on and add that as a feature in our data.

Fourier Transform Features We add frequency and amplitude to our feature set following [6, 11, 12]. They state that any repeating pattern can be decomposed as a sum of multiple sine waves, each sine wave can be represented with a frequency and amplitude. Frequency being oscillations per minute for the wave and amplitude being the significance of the frequency compared to other frequencies. As we see in Figure11(a), our pickups for a cluster follows a pattern in the 24 hour time period and thus can be decomposed into multiple sine waves. Different clusters will have different patterns. For each cluster, we take the 5 frequencies with the highest amplitudes and add them as features in our data.

weather data Finally we added the weather data we acquired earlier as a feature. We hypothesize that the weather conditions play a vital role in taxi demands going up or down. For each cluster center, we find the nearest location from our weather data and assign the information from that location to our clusters. The weather data had some NAN values which we replaced with 0.



(a)Pickup pattern for Cluster 1 (b) Frequency and amplitude of Fourier transformed wave

Fig. 11.

Final Features Finally we have 33 features which are listed in Table 1,

Preparing Labels The true amount of pickups per time slot per cluster are our labels. Since it's hard to exactly predict the value for pickups and calculating the error will be tricky, we changed the pickup values to the nearest number which is a multiple of 50. For example, 202 pickups will be changed to 200, 238 will be changed to 250.

3.6 Prediction

Separating training and testing dataset Since our problem is a time series problem, we can't split the data randomly, it was to be split on the basis of time. The full data was sorted based on time, then we allocated 80% for training and 20% for testing.

Models Since our problem falls in the time series forecasting criteria which is a kind of regression problem, we used three different regression algorithms and a neural network based classifier and compared their performances. We wanted to test if the neural network can outperform traditional methods.

The models we used are

- Linear Regression
- Random Forest Regression
- XGBoost Regression
- Multi Level Perception(MLP) Classifier(Neural Network)

For the traditional approaches, we used Grid Search to tune the parameters. For the the neural network, we used ***adam*** optimizer with (150, 150) hidden layers.

feature(1-5)	The number of pickups from last 5 time slots
average	Weighted moving average of pickups from last 5 time slots
frequency and amplitude(1-5)	5 highest amplitudes of Fourier transformed wave and the frequencies corresponding to them
cluster center	latitude and longitude of the cluster center
time slot	Unique time slot number
weekday	Day of the week of pickup
AWND	Monthly average wind speed in tenths of meters/second
EMNT	Extreme minimum temperature for the month in tenths of degree Celsius
EMSD	Extreme maximum snow depth in millimeters
EMSN	Highest daily snowfall in the month in millimeters
EMXP	Highest daily total of precipitation in the month in tenths of millimeters
EMXT	Extreme maximum temperature for the month in tenths of degree Celsius
HDSD	Running total of monthly Heating Degree Day through the end of the most recent month.
PRCP	Total Monthly precipitation in millimeters to tenths.
SNOW	Total Monthly Snowfall in millimeters
TAVG	Average Monthly Temperature in degree Celcius
TMAX	Average of the daily maximum temperatures
TMIN	Average of the daily minimum temperatures
TSUN	Daily total sunshine in minutes

Table 1. Features

Model	Datatype	Train MAPE(%)	Train MSE	Test MAPE(%)	Test MSE
Linear Regression	Yellow Cab	13.355924	185.022980	14.397238	272.951709
Random Forest Regression	Yellow Cab	**4.753929**	**23.701961**	**13.096294**	**202.902015**
XGBoost Regression	Yellow Cab	11.713362	137.883731	14.536970	242.794858
Multi Level Perception	Yellow Cab	13.563219	198.176385	15.137497	296.043386
Linear Regression	Green Cab	33.774125	8.294483	**32.578688**	**9.379364**
Random Forest Regression	Green Cab	**12.569301**	**1.142423**	37.664249	10.659669
XGBoost Regression	Green Cab	31.495877	6.901285	37.077264	10.525930
Multi Level Perception	Green Cab	33.819814	8.956077	33.741095	10.658240

Table 2. Results

Performance Metric To evaluate the perform ace of the prediction we used Mean Absolute Percentage Error(MAPE) and Mean Squared Error(MSE) which is a well known prediction of measure accuracy for forecasting problem. The formulas is given in eq 6 and eq 7

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \frac{Actual\,Value_t - Predicted\,Value_t}{Actual\,Value_t} \tag{6}$$

$$MSE = \frac{1}{n} \sum_{t=1}^{n} Actual\,Value_t - Predicted\,Value_t{}^2 \tag{7}$$

MAPE is a measure of relative or percentage error and MSE measures absolute error. A relative error of 5% might mean the predicted value was 525 whereas the actual value was 500. But an absolute error of 5 means the the predicted value was 505 or 495 for an actual value of 500.

Results Table 2 shows our experimental results for both yellow and green cab data. Random Forest Regression had the best results for yellow cab data. But we can see there is a large difference between train and test losses, which means the model might be over fitting. For green cabs, Random Forest Regression had the best train loss, but Linear Regression had the best test loss.

Our MLP model has comparable losses, but it loses by a slight margin to other models. It even outperformed Random Forest and XGBoost Regression models for green cab data. We might be able to get results for MLP by tuning the parameters more.

4 Conclusions and Future Works

In this work, we explore the performances of various regression models and a MLP neural network model in predicting taxi demand in a time slot based on location information and demand for taxis in that location in the past. In future, we would like to identify more significant features so that we can improve the accuracy of prediction and minimize loss. The weather data we used was the average of the whole month for a particular location, we would like to try and get weather data by day or hour for a location as feature and see if that increases our accuracy. We can also try different averaging techniques like exponential moving average instead weighted moving average. Since we are training multiple models, we can also try adding predictions from one models as training features for the next one, We could also try the ensemble approach, training the models in parallel and combining their predictions.

References

1. Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., Damas, L. (2013). Predicting Taxi–Passenger Demand Using Streaming Data. IEEE Transactions on Intelligent Transportation Systems, 14(3), 1393–1402. https://doi.org/10.1109/tits.2013.2262376
2. Zhao, K., Khryashchev, D., Freire, J., Silva, C., Vo, H. (2016). Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. 2016 IEEE International Conference on Big Data (Big Data). Presented at the 2016 IEEE International Conference on Big Data (Big Data). https://doi.org/10.1109/bigdata.2016.7840676
3. Xu, J., Rahmatizadeh, R., Boloni, L., Turgut, D. (2018). Real-Time Prediction of Taxi Demand Using Recurrent Neural Networks. IEEE Transactions on Intelligent Transportation Systems, 19(8), 2572–2581. https://doi.org/10.1109/tits.2017.2755684
4. Wong, K. I., Wong, S. C., Yang, H. (2001). Modeling urban taxi services in congested road networks with elastic demand. Transportation Research Part B: Methodological, 35(9), 819–842. https://doi.org/10.1016/s0191-2615(00)00021-7
5. Yang, H., Lau, Y. W., Wong, S. C., Lo, H. K. (2000). Transportation, 27(3), 317–340. https://doi.org/10.1023/a:1005289504549
6. J. Deri,J. Moura. (2015). Taxi data in New York city: A network perspective. 1829-1833. 10.1109/ACSSC.2015.7421468.
7. TLC Trip Record Data, https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
8. 2016 New York City Taxi Zones, $https://geo.nyu.edu/catalog/nyu_2451_36743$
9. Global Weather Summary of the Month, https://www.ncdc.noaa.gov/cdo-web/datasets/GSOM/locations/CITY:US360019/detail
10. Unix Time, $https://en.wikipedia.org/wiki/Unix_time$
11. Taxi Demand Prediction- New York City, https://blog.goodaudience.com/taxi-demand-prediction-new-york-city-5e7b12305475
12. Taxi demand prediction in New York City, https://medium.com/@ranasinghiitkgp/taxi-demand-prediction-in-new-york-city-916cde6a3492