JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

UBER PICKUP DATA ANALYSIS

INDEX

INTRODUCTION

Talking about our Uber data analysis project, data analysis is an important component of *Machine Learning* through which companies are able to understand the background of various operations. With the help of visualization, companies can avail the benefit of understanding the complex data and gain insights that would help them to craft decisions.

REVIEW/BACKGROUND MATERIAL

For our project we have taken the New York city datasets(April 2014- Sept 2014) from Kaggle and performed various data manipulation techniques to make them fit for visualization and time series analysis. We then analyze the dataset to see in which location do we have maximum cabs.
To be able to complete this project we first learned about the python language and installed many files on our software system (Visual Studio Code). We referred to many videos and study material that are given in the ***Reference*** section.

DESCRIPTION AND RESULTS

Preprocessing.ipynb

The dataset was loaded and merged, in order to create a new dataset with 4,534,327 rows in the dataset. All this information was used to group by and count the number of pickups pertaining to a specified time period.

The initial dataset had 4 attributes – Date/Time of pickup, latitude, longitude of pickup and base number.

```
dataset.head() # This gives us the schema of the dataset
✓ 0.7s
```

	Date/Time	Lat	Lon	Base
0	4/1/2014 0:11:00	40.7690	-73.9549	B02512
1	4/1/2014 0:17:00	40.7267	-74.0345	B02512
2	4/1/2014 0:21:00	40.7316	-73.9873	B02512
3	4/1/2014 0:28:00	40.7588	-73.9776	B02512
4	4/1/2014 0:33:00	40.7594	-73.9722	B02512

The Date/Time column was converted from string to multiple attributes of numeric type.

```
dataset.head()
✓ 0.5s
```

	Date/Time	Lat	Lon	Base	Month	Day	Day-of-week	Day-of-week-num	hours	minutes
0	4/1/2014 0:11:00	40.7690	-73.9549	B02512	4	1	Tuesday	1	0	11
1	4/1/2014 0:17:00	40.7267	-74.0345	B02512	4	1	Tuesday	1	0	17
2	4/1/2014 0:21:00	40.7316	-73.9873	B02512	4	1	Tuesday	1	0	21
3	4/1/2014 0:28:00	40.7588	-73.9776	B02512	4	1	Tuesday	1	0	28
4	4/1/2014 0:33:00	40.7594	-73.9722	B02512	4	1	Tuesday	1	0	33

All the redundant columns were removed.

```
dataset.head()
✓ 0.7s
```

	Lat	Lon	Base	Month	Day	Day-of-week	Day-of-week-num	hours	minutes
0	40.7690	-73.9549	B02512	4	1	Tuesday	1	0	11
1	40.7267	-74.0345	B02512	4	1	Tuesday	1	0	17
2	40.7316	-73.9873	B02512	4	1	Tuesday	1	0	21
3	40.7588	-73.9776	B02512	4	1	Tuesday	1	0	28
4	40.7594	-73.9722	B02512	4	1	Tuesday	1	0	33

And then the dataset was checked for NULL values.

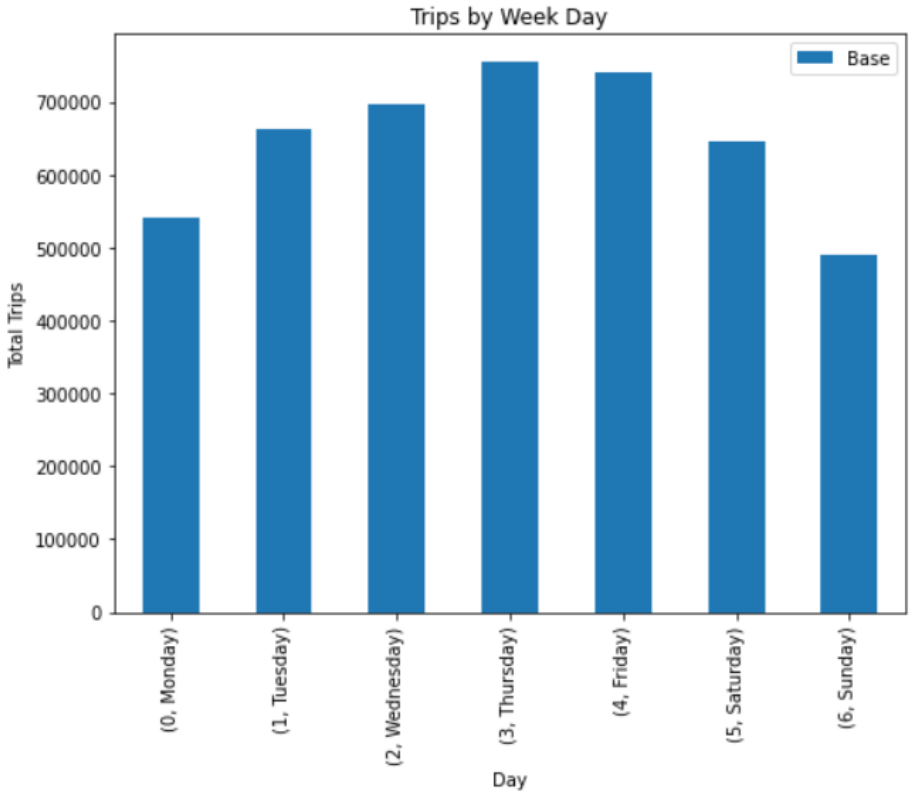
```
   dataset.describe(), dataset.shape
✓ 1.3s
(                Lat           Lon         Month           Day  \
 count  4.534327e+06  4.534327e+06  4.534327e+06  4.534327e+06
 mean   4.073926e+01 -7.397302e+01  6.828703e+00  1.594337e+01
 std    3.994991e-02  5.726670e-02  1.703810e+00  8.744902e+00
 min    3.965690e+01 -7.492900e+01  4.000000e+00  1.000000e+00
 25%    4.072110e+01 -7.399650e+01  5.000000e+00  9.000000e+00
 50%    4.074220e+01 -7.398340e+01  7.000000e+00  1.600000e+01
 75%    4.076100e+01 -7.396530e+01  8.000000e+00  2.300000e+01
 max    4.211660e+01 -7.206660e+01  9.000000e+00  3.100000e+01

        Day-of-week-num         hours       minutes
 count     4.534327e+06  4.534327e+06  4.534327e+06
 mean      2.968115e+00  1.421831e+01  2.940071e+01
 std       1.875971e+00  5.958759e+00  1.732238e+01
 min       0.000000e+00  0.000000e+00  0.000000e+00
 25%       1.000000e+00  1.000000e+01  1.400000e+01
 50%       3.000000e+00  1.500000e+01  2.900000e+01
 75%       5.000000e+00  1.900000e+01  4.400000e+01
 max       6.000000e+00  2.300000e+01  5.900000e+01  ,
 (4534327, 9))


here are no NULL values in the dataset. The dataset has 4,534,327 rows in the dataset
```

Visualization.ipynb

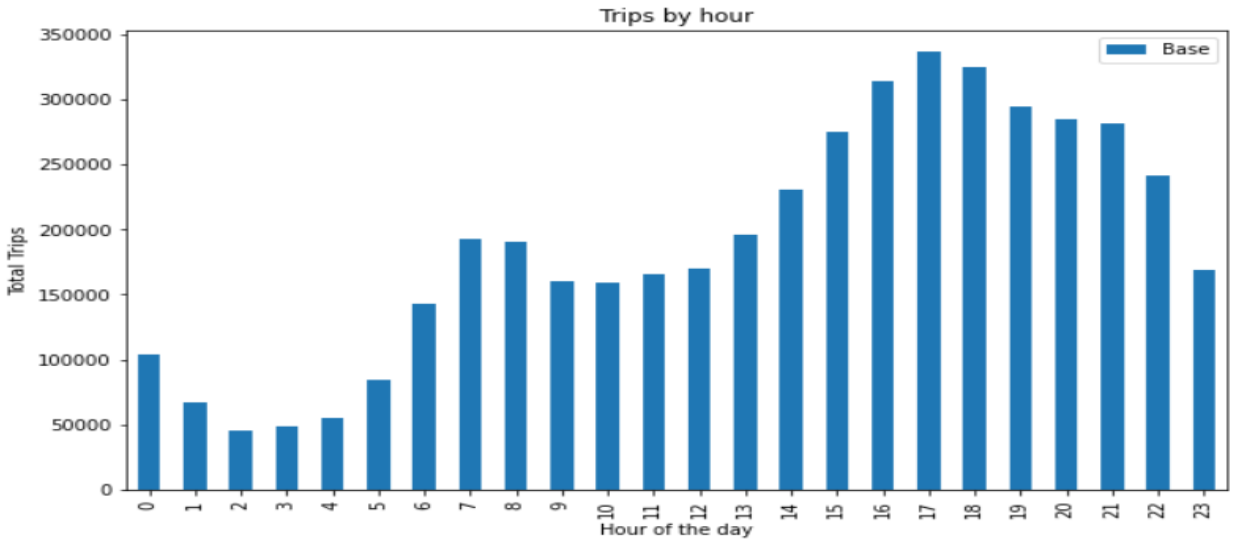
Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed. Python offers multiple great graphing libraries that come packed with lots of different features. Here are some of the plotting libraries that we used in our project:

- Matplotlib
- Pandas Visualization
- Seaborn

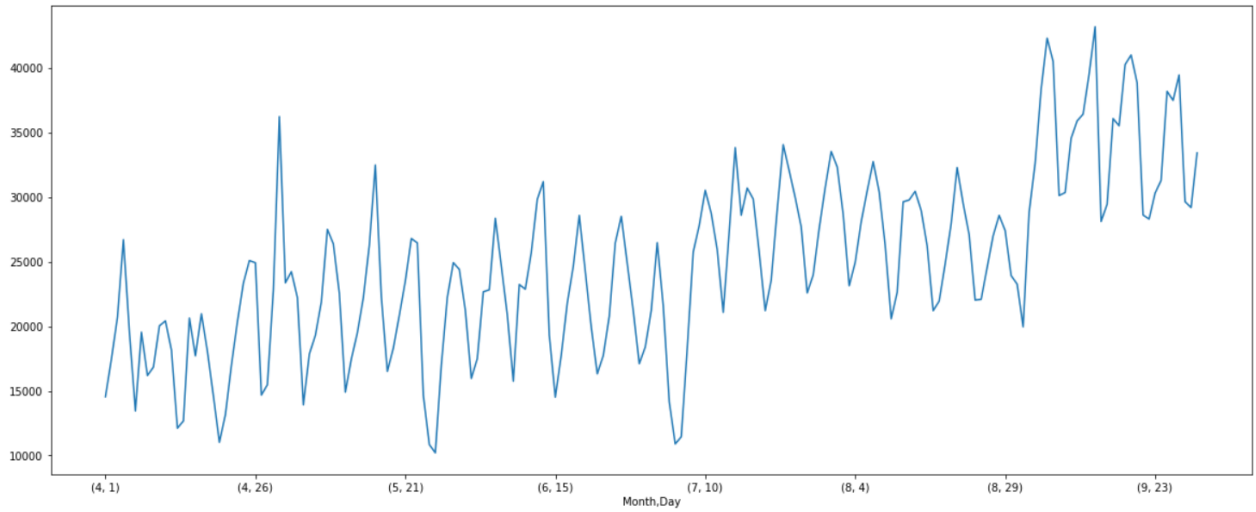
Graph with the total number of trips made per day from April(2014) – September(2014).



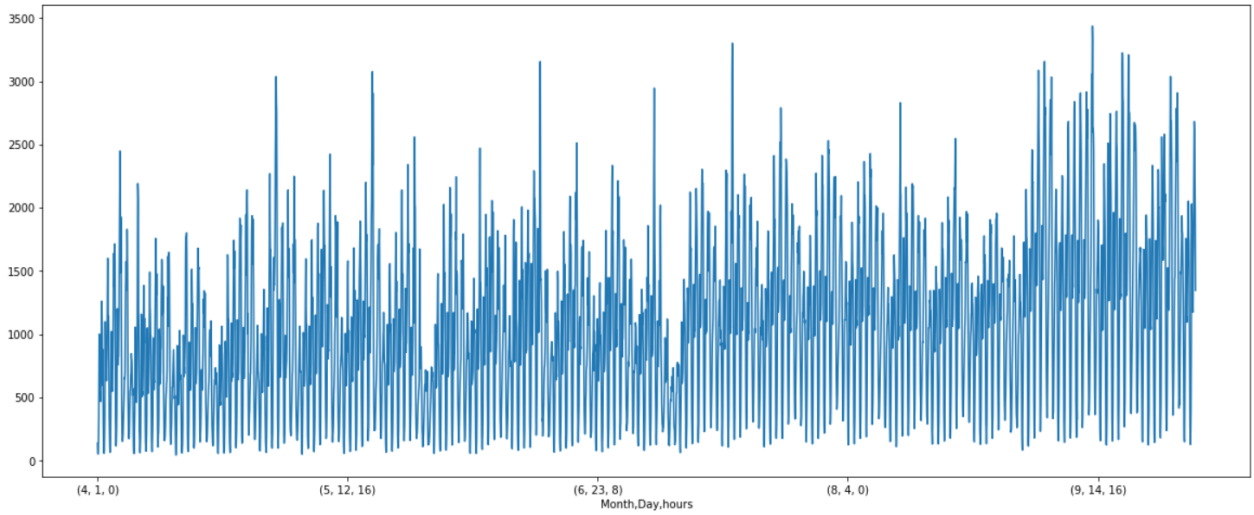
Graph with a total number of trips made hourly each day from April(2014) – September(2014).



Dataframe(I) of number of trips on all days from April 2014 to September 2014



Dataframe(II) of number of trips made hourly each day from April 2014 to September 2014

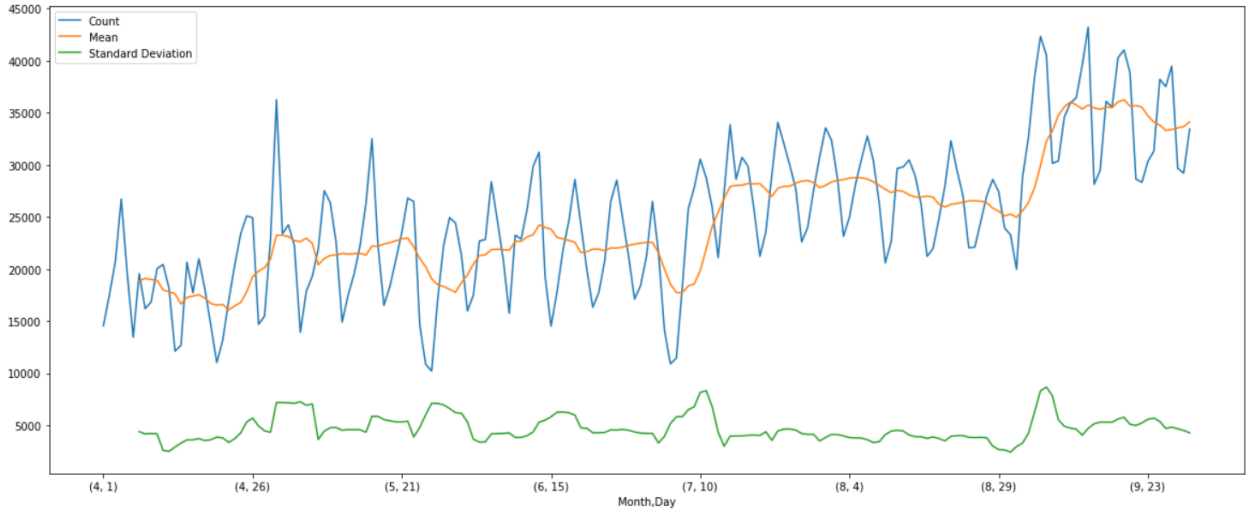


On both the dataframes we performed two stationarity test:

1) Plotting the rolling mean and rolling standard deviation.
2) Performing Augmented Dicky-Fuller (ADF) test.

In ADF test there is a hypothesis testing involved with a null and alternate hypothesis and as a result a test statistic is computed and p-values get reported. If p-value < 0.05the time series is stationary.

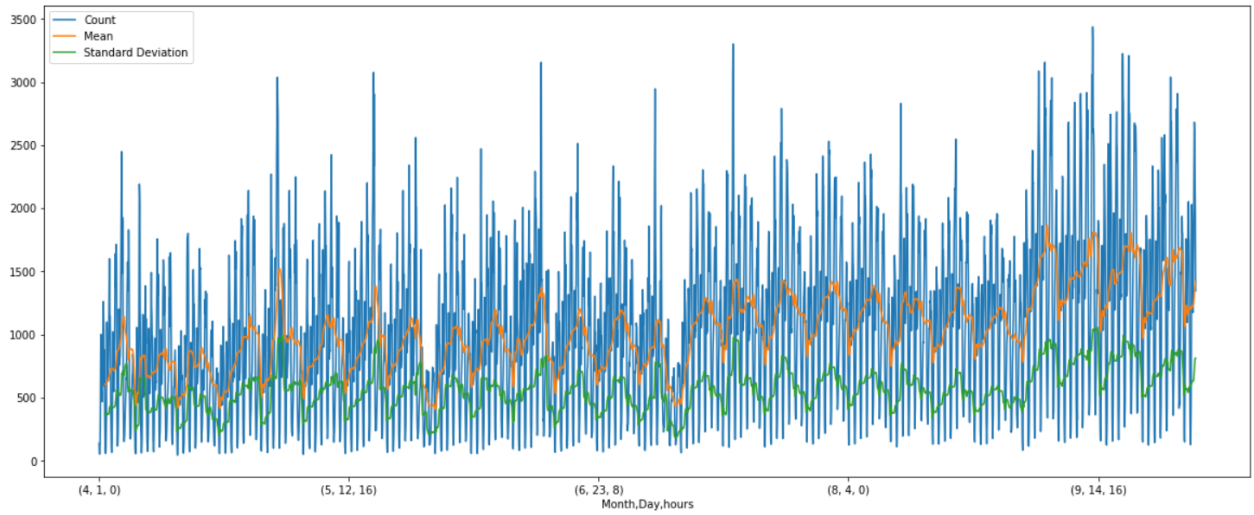
Graph of rolling mean and rolling standard deviation with a window of 7 days(for dataframe I)




```
ADF Statistic: -0.703280
p-value: 0.845898
Critical Values:
        1%: -3.469
        5%: -2.879
        10%: -2.576
```

As one can see the graph doesn't appear to be stationary and also the p-value>0.05. The Month,Day dataframe(I) is not stationary.

Graph of rolling mean and rolling standard deviation with a window of 7*24 hours(for dataframe II)



The graph for dataframe II is inconclusive.

So we performed ADF test,

```
ADF Statistic: -4.906253
p-value: 0.000034
Critical Values:
        1%: -3.432
        5%: -2.862
        10%: -2.567
```

As the p-value<0.05 the Month,Day,hour dataframe is stationary. And only this dataframe is fit for TimeSeries analysis.

TimeSeries.ipynb

Time series is a sequence of observations recorded at regular time intervals. Depending on the frequency of observations, a time series may typically be hourly, daily, weekly, monthly, quarterly and annual. Why even analyze a time series? Because it is the preparatory step before you develop a forecast of the series. So what does analyzing a time series involve? Time series

analysis involves understanding various aspects about the inherent nature of the series so that you are better informed to create meaningful and accurate forecasts.
Forecasting is the process of making predictions based on past and present data and most commonly by analysis of trends.

The dataset was read and stored in a variable called data. Then a column was added to add year to the data and a new column called time was added which contained the hours,data,month and year in date-time format.
The data was manipulated to contain only two columns - time and rides.

```
# Print the first 5 values to see the schema of the dataframe
data.head()
```

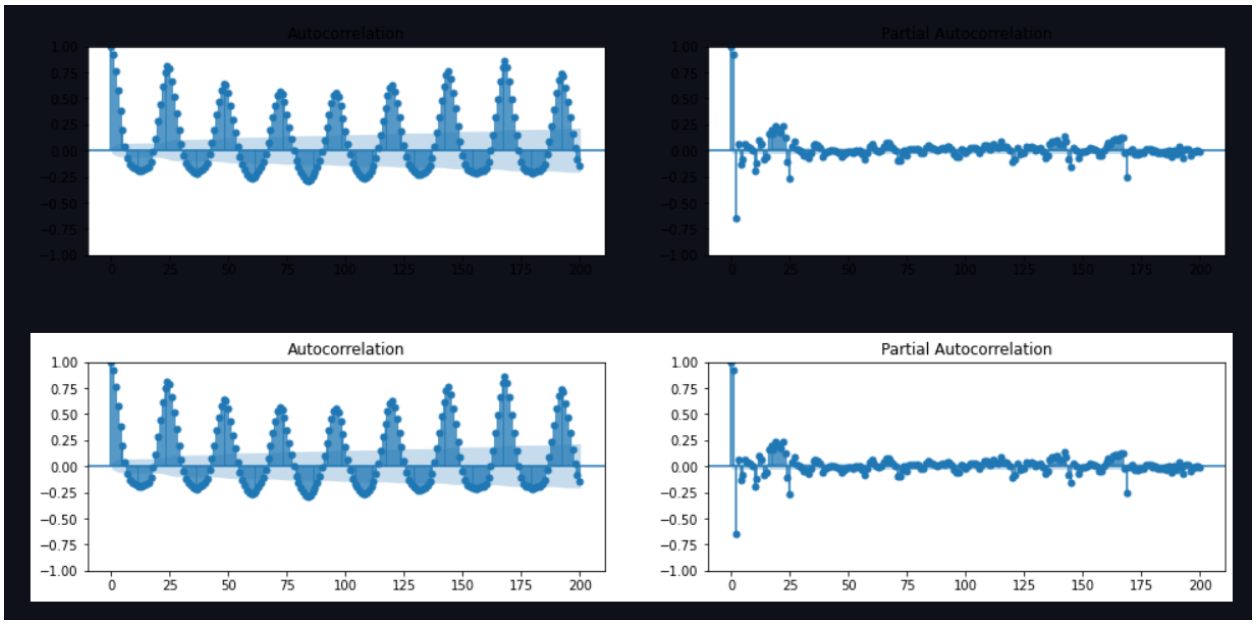
✓ 0.7s

	time	rides
0	2014-04-01 00:00:00	138
1	2014-04-01 01:00:00	66
2	2014-04-01 02:00:00	53
3	2014-04-01 03:00:00	93
4	2014-04-01 04:00:00	166

Then, the plots for autocorrelation and partial autocorrelation were plotted.
Autocorrelation refers to how correlated a time series is with its past values whereas the ACF is the plot used to see the correlation between the points, up to and including the lag unit.

An *autocorrelation* (ACF) plot represents the autocorrelation of the series with lags of itself. In ACF, the correlation coefficient is in the x-axis whereas the number of lags is shown in the y-axis.
A *partial autocorrelation* (PACF) plot represents the amount of correlation between a series and a lag of itself that is not explained by correlations at all lower-order lags.

From the graph, we can see that there is a strong positive relation between values every 24 hours. Also, we can see that the strongest correlation is at value 168 i.e. 24x7 (exactly 1 week apart).



We then split the data into training and testing dataframes. Splitting them as follows :

```
train = data.iloc[: 24 * -30]
test = data.iloc[24 * -30 : ]
```

The exponential smoothing approach was used to predict the future values of the number of trips. We used the library function for this calculation.

The parameters used here were:

np.array(train[‘rides’]) : This takes the 'rides' column from the dataframe, and converts it into a numpy array, for easier calculation

seasonal_periods: The number of periods in a complete seasonal cycle, e.g., 4 for quarterly data or 7 for daily data with a weekly cycle. 240 is picked by trial and error

trend : The type of trend component. Add means that trend adds on to the seasonal part, and not multiplied by it.

seasonal : Same as trend

calling the fit function will fit the training data to some parameters, and thus help us get the predictions

model.forecast will forecast the value of the time series

Converting the numpy array to a time series dataframe, with same dates as test data frame and displaying it.

```
prediction.head()
✓ 0.3s
```

	time	prediction
0	2014-08-31 23:00:00	1020.663841
1	2014-09-01 00:00:00	794.432113
2	2014-09-01 01:00:00	620.727674
3	2014-09-01 02:00:00	543.780138
4	2014-09-01 03:00:00	564.462598

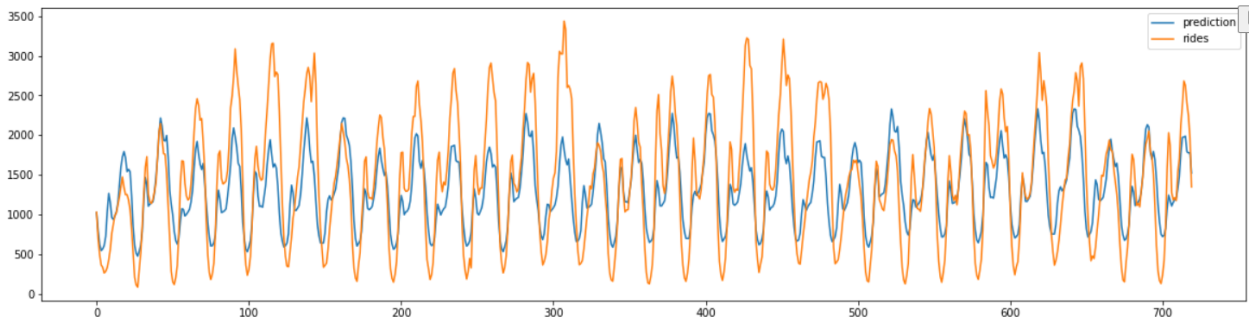
Smoothing methods work as weighted averages. Forecasts are weighted averages of past observations. The weights can be uniform (this is a moving average), or following an exponential decay — this means giving more weight to recent observations and less weight to old observations. More advanced methods include other parts in the forecast, like seasonal components and trend components.

Exponential smoothing is a rule of thumb technique for smoothing time series data using the exponential window function. Whereas in the simple moving average the past observations are weighted equally, exponential functions are used to assign exponentially decreasing weights over time.

We used Holt-Winters approach. Holt's LES model addresses this issue by including *two* smoothing constants, one for the level and one for the trend. At any time t, as in Brown's model, there is an estimate L_t of the local level and an estimate T_t of the local trend. Here they are

computed recursively from the value of Y observed at time t and the previous estimates of the level and trend by two equations that apply exponential smoothing to them separately.

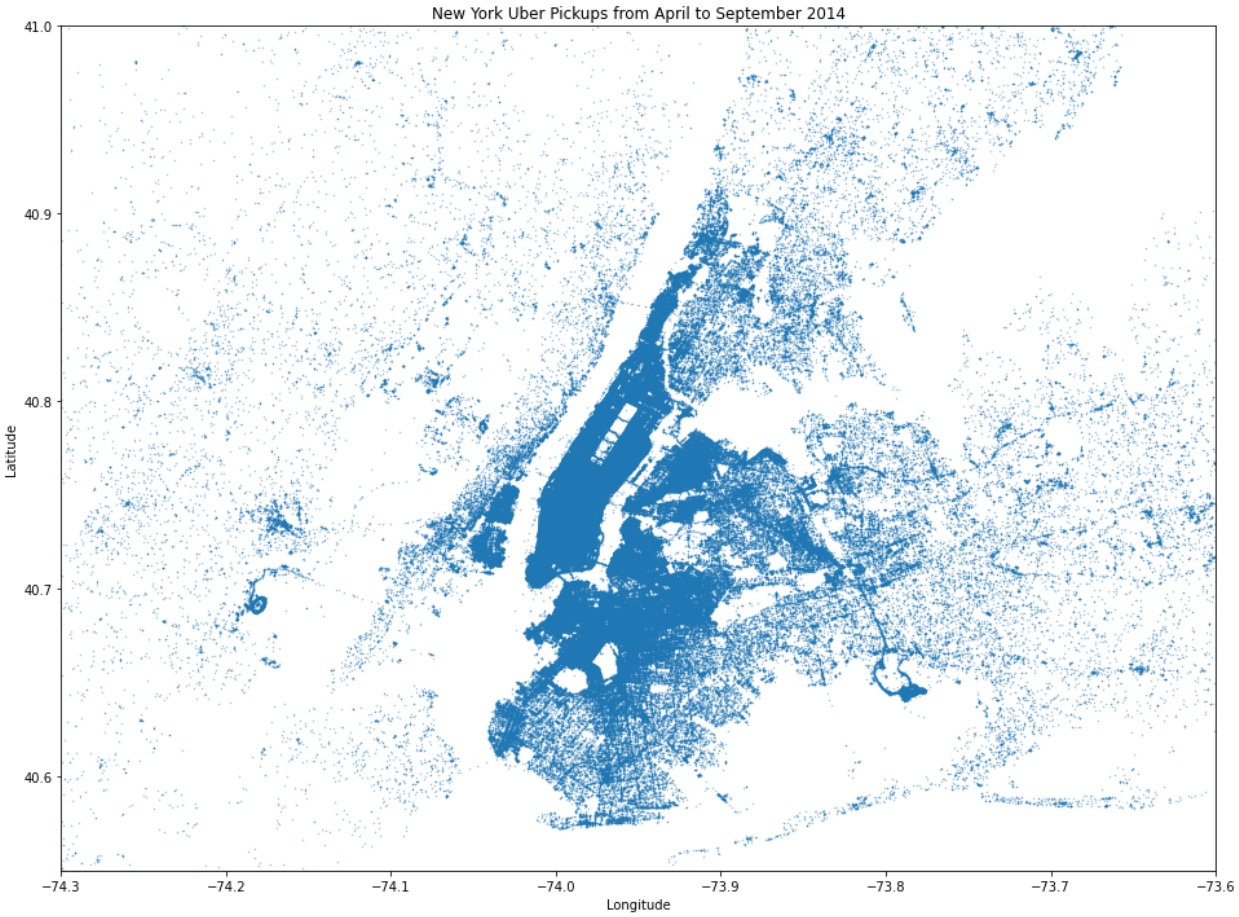
Here, we plotted the predicted values against the testing data.
It can be seen that exponential smoothing encapsulates the basic trends perfectly.



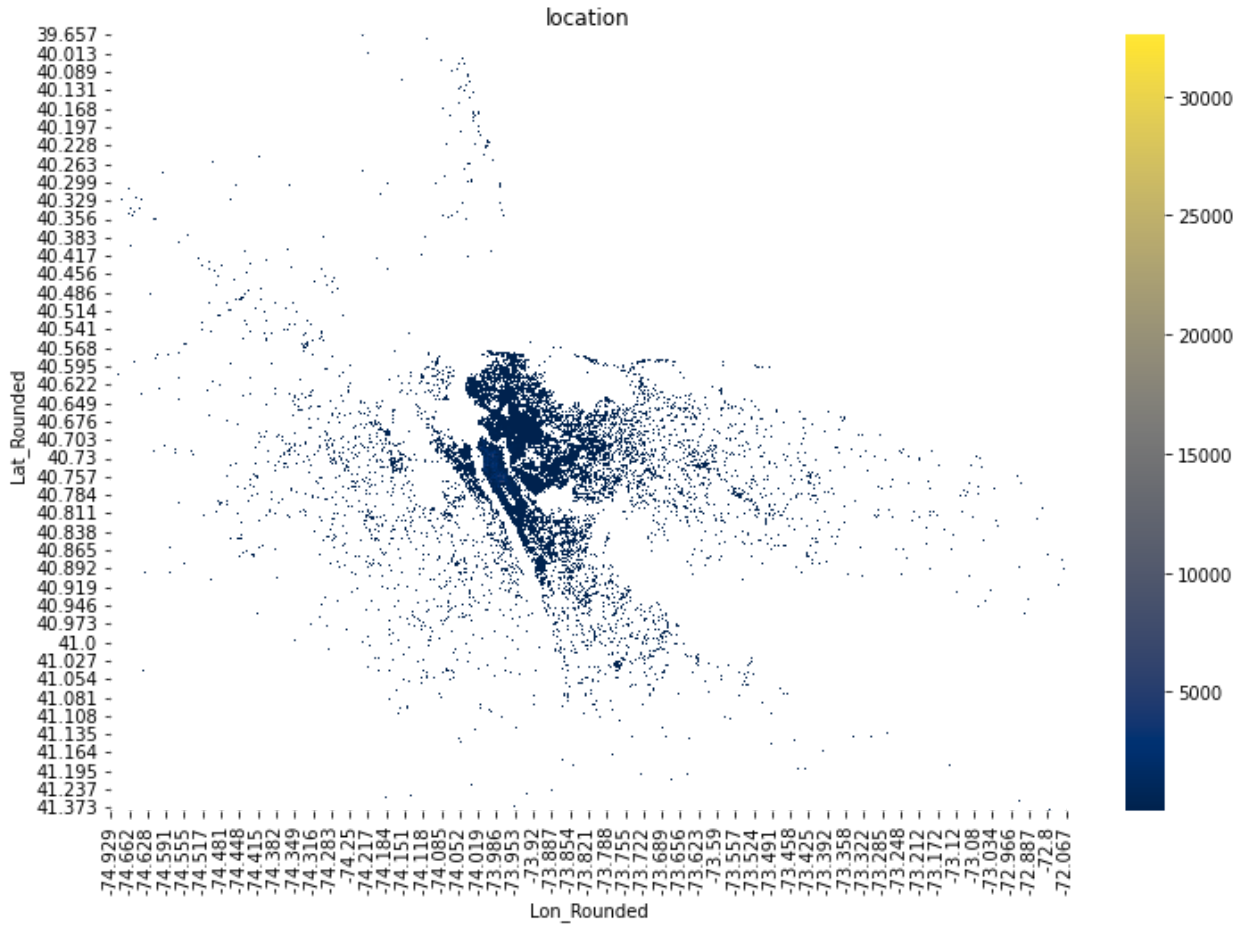
Geolocation.ipynb

A heatmap contains values representing various shades of the same color for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different color can also be used.
Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function.

The figure below is a heatmap of total uber pickups in New York in 2014 according to their latitude and longitude.



The latitude and longitude in the dataset have been reduced to three decimal places to plot a more proper and visible heatmap.



The data is then divided according to different time ranges i.e. 0-6, 6-12,12-18 and 18-24.

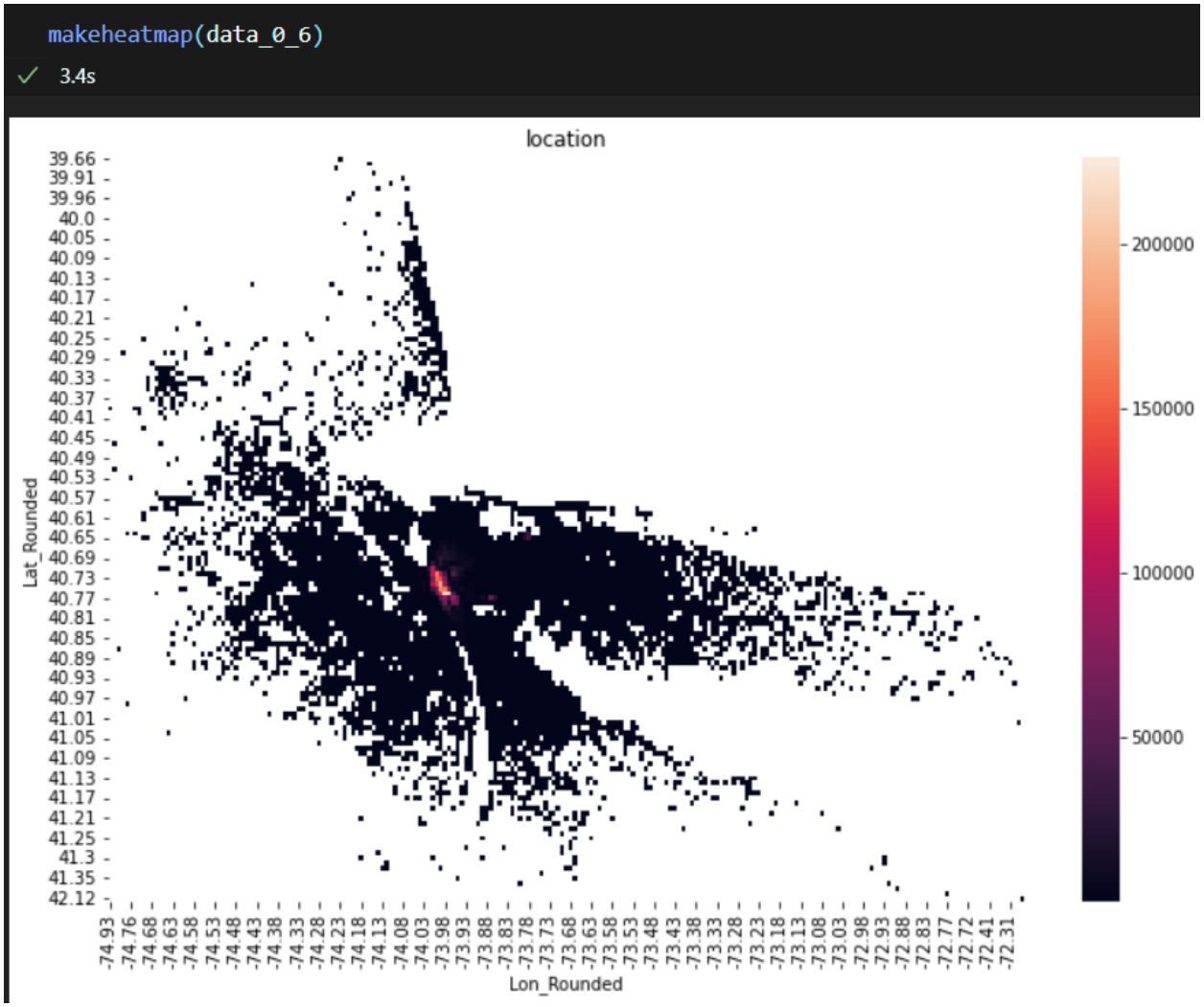
Then, the total number of rides in the respective time ranges is calculated.

```
data_0_6 = dataset.loc[ dataset["hours"].isin([0, 1, 2, 3, 4, 5]) ]
data_6_12 = dataset.loc[ dataset["hours"].isin([6, 7, 8, 9, 10, 11]) ]
data_12_18 = dataset.loc[ dataset["hours"].isin([12, 13, 14, 15, 16, 17]) ]
data_18_0 = dataset.loc[ dataset["hours"].isin([18, 19, 20, 21, 22, 23]) ]
len(data_0_6), len(data_6_12), len(data_12_18), len(data_18_0)
```

```
(404384, 1011629, 1522010, 1596304)
```

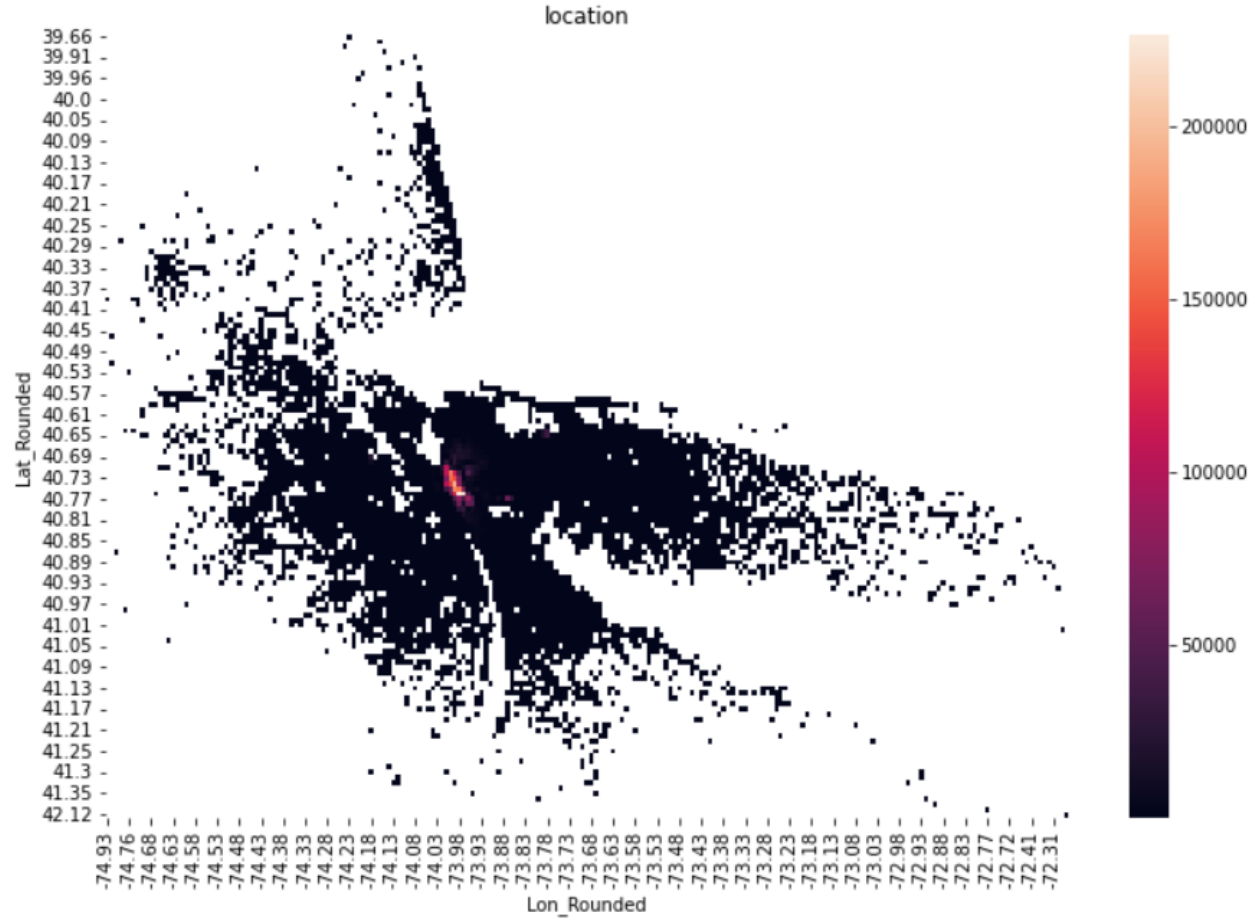
Then, we created a function ‘makeheatmap’ and used it to plot the different datasets which were previously divided according to time range.

The heatmap given below shows the location of uber pickups done between 00 to 06 hours.



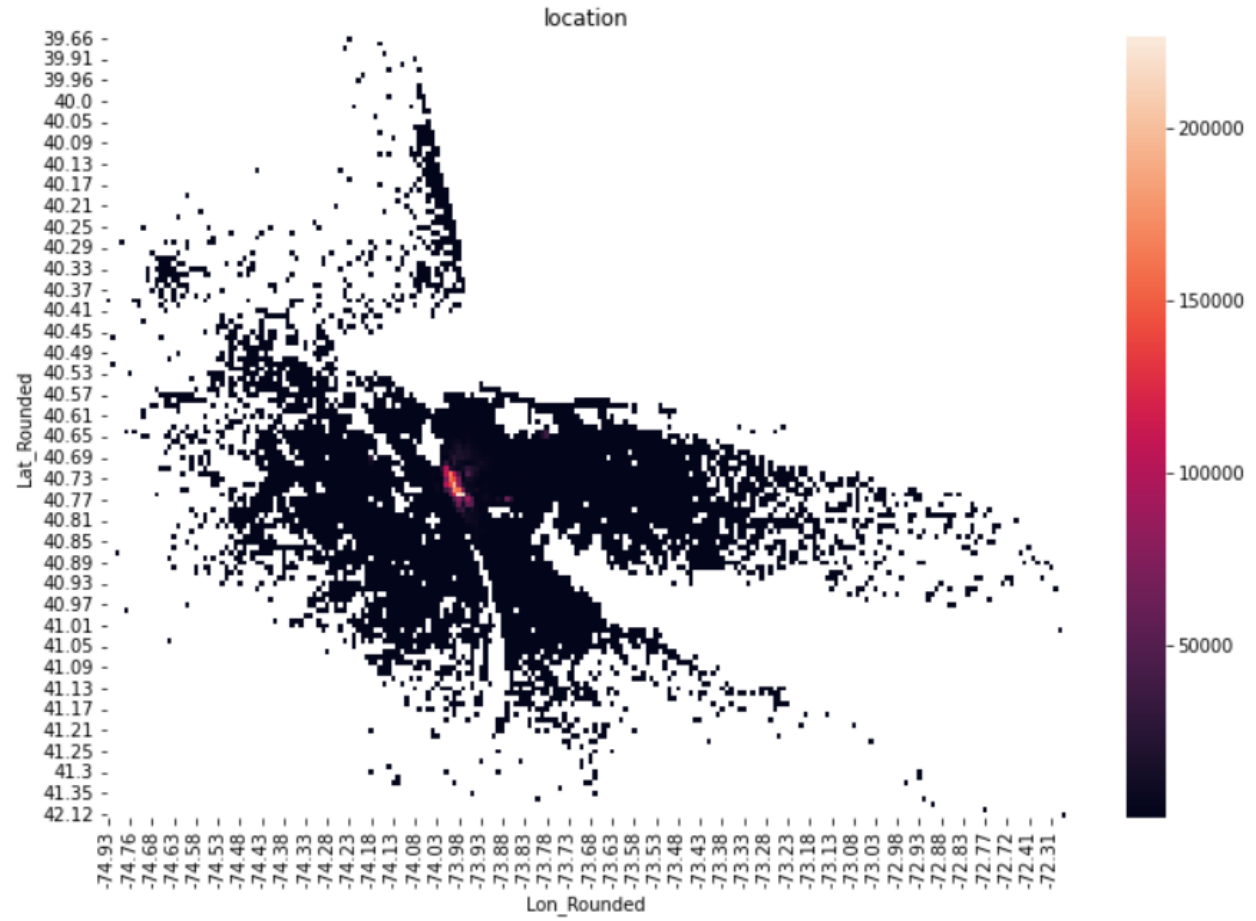
The heatmap given below shows the location of uber pickups done between 06 to 12 hours.

```
makeheatmap(data_6_12)
✓ 2.6s
```

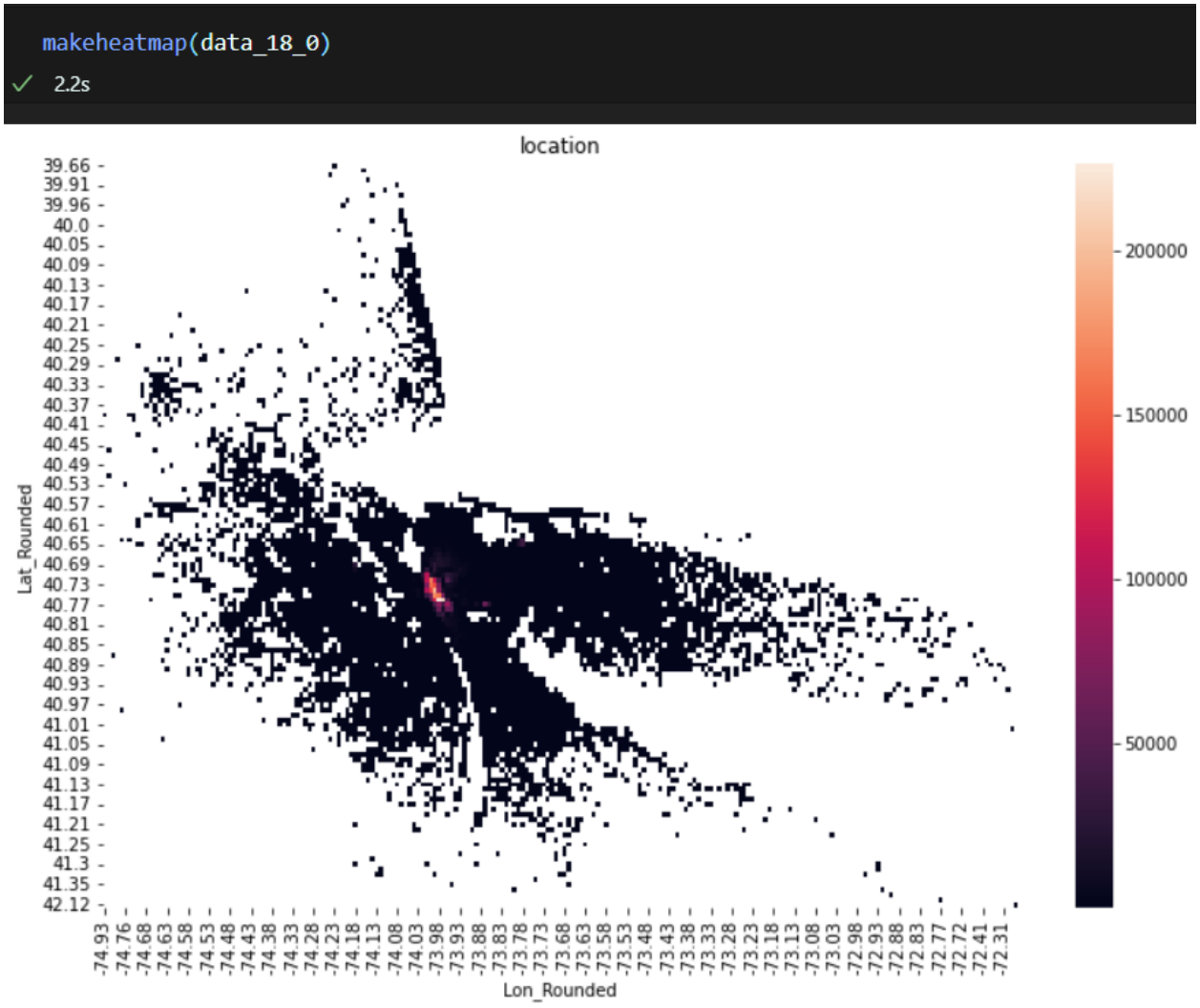


The heatmap given below shows the location of uber pickups done between 12 to 18 hours.

```
makeheatmap(data_12_18)
✓ 2.6s
```

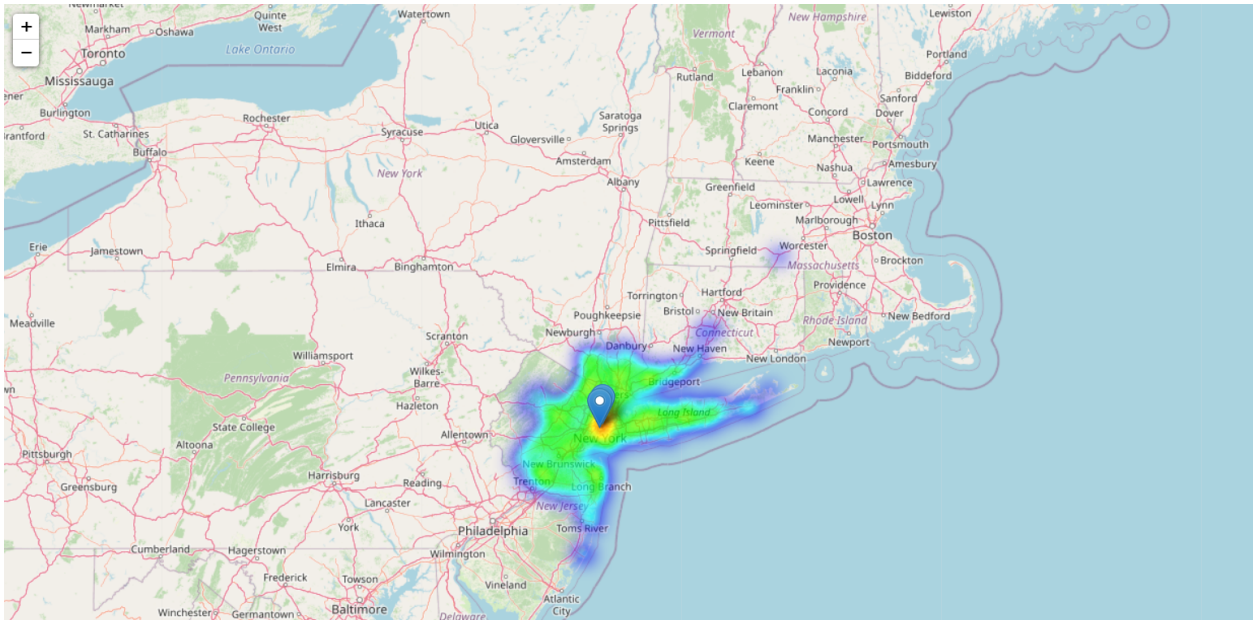


The heatmap given below shows the location of uber pickups done between 18 to 00 hours.

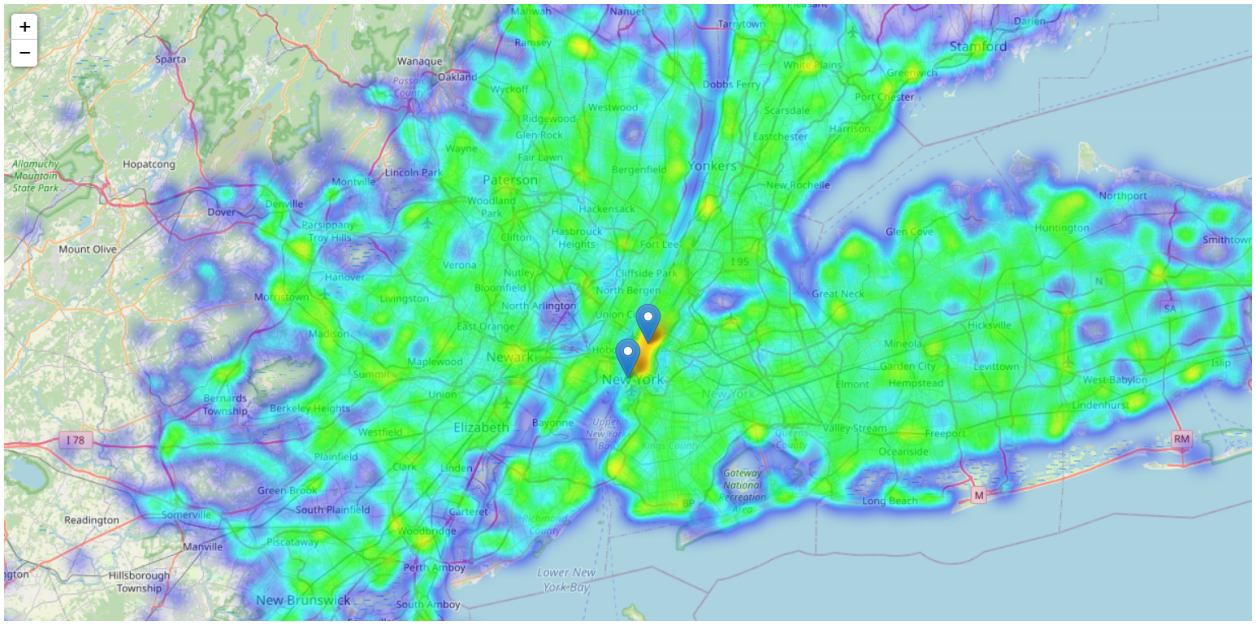


Even though the graphs showed similar results, the number of rides varied in different time ranges.

Here, the heatmap has been plotted on a geographical map (using folium). As we can observe, the density of the rides is high near the Manhattan area.
We also marked the locations of the One World Trade Center and The Empire State Building which are popular hotspots for uber pickups, which is also evident from the maps.

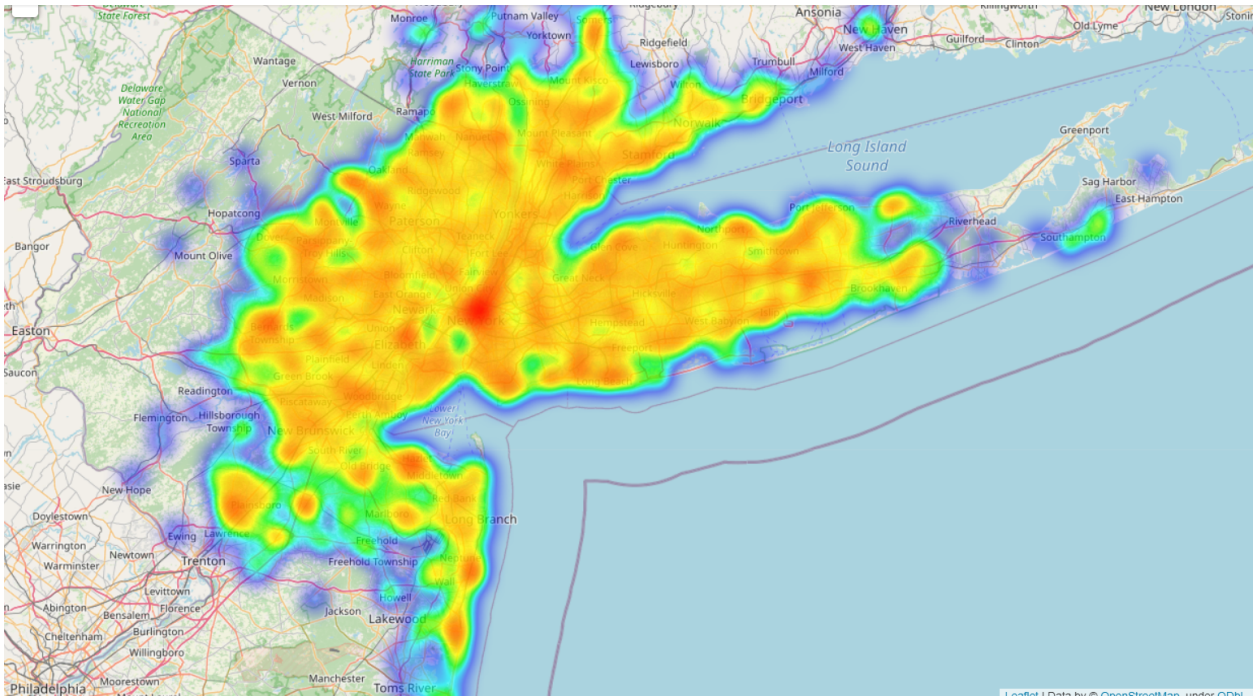


Geographical map showing Uber pickups in New York



Uber pickup density along with location pins at One World Trade Center and Empire State Building

The map after reducing the "Influence" of each point on the heatmap by using a weight of 0.5 (by default it is 1)



CONCLUSION

So this is how we can analyze the Uber trips for New York City. Some of the conclusions that we got from this analysis are:

- The number of rides booked in a week steadily increases and dips down on weekends. This is consistent with our common knowledge of the work week.
- Thursday is the most profitable day for Uber.
- On Sundays, fewer people use Uber.
- We can see a small spike in the number of uber rides around 9 AM, and the peak traffic is at 5 PM, which is commonly known as the rush hour. Early morning hours see very little traffic.
- 5 pm is the busiest hour of the day for Uber.
- On average, a rise in Uber trips starts around 6 am.
- New York stands up to its reputation of being called “the City that Never Sleeps” as we can see a good number of uber pickups throughout the day and throughout the city.
- Most of the Uber trips originate near the Manhattan region in New York as evident from the heatmap.
- In the late hours of the day (18-24 hours), the number of rides is the maximum according to the heatmap.
- We also marked the locations of the One World Trade Center and The Empire State Building which are popular hotspots for uber pickups, which is also evident from the maps.

REFERENCES

[1] Eva Ostertagová, Oskar Ostertag, "The Simple Exponential Smoothing Model" , Modelling Of Mechanical and Mechatronic Systems, 2011, Technical university of Košice.

[2] Rostislav Netek, Tomas Pour, and Renata Slezakova, "Implementation of Heat Maps in Geographical Information System – Exploratory Study on Traffic Accident Data" ,From the journal Open Geosciences.

[3] Shilin Zhao, Yan Guo, Quanhu Sheng, and Yu Shyr, "Advanced Heat Map and Clustering Analysis Using Heatmap3", Center for Quantitative Sciences, Vanderbilt University, Nashville, TN 37232, USA.

[4] Md. Habibur Rahman, Umma Salma, Md. Moyazzem Hossain,
Md. Tareq Ferdous Khan,Revenue Forecasting using Holt–Winters
Exponential Smoothing, Department of Statistics, Jahangirnagar University, Dhaka, Bangladesh.

[5] Accessed Date [17.09.21] URL:
https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

[6] Accessed Date [02.10.21] URL:
https://towardsdatascience.com/time-series-modeling-using-scikit-pandas-and-numpy-682e3b8db8d1

[7] Accessed Date [14.10.21] URL:
https://medium.com/analytics-vidhya/time-series-forecasting-sarima-vs-auto-arima-models-f95e76d71d8f

[8] Accessed Date [07.10.21] URL: https://www.machinelearningplus.com/time-series/time-series-analysis-python/

[9] Accessed Date [09.11.21] URL: https://towardsdatascience.com/time-series-in-python-exponential-smoothing-and-arima-processes-2c67f2a52788

[10] Accessed Date [19.11.21] URL: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

[11] Accessed Date [24.11.21] URL: https://machinelearningmastery.com/exponential-smoothing-for-time-series-forecasting-in-python/

[12] Accessed Date [2.12.21] URL: https://analyticsindiamag.com/hands-on-guide-to-time-series-analysis-using-simple-exponential-smoothing-in-python/

[13] Accessed Date [3.12.21] URL: https://www.statsmodels.org/dev/examples/notebooks/generated/exponential_smoothing.html

[14] Accessed Date [4.12.21] URL: https://towardsdatascience.com/heatmap-basics-with-pythons-seaborn-fb92ea280a6c

[15] Accessed Date [5.12.21] URL:https://towardsdatascience.com/data-101s-spatial-visualizations-and-analysis-in-python-with-folium-39730da2adf

[16] Accessed Date [5.12.21] URL: https://www.analyticsvidhya.com/blog/2020/06/guide-geospatial-analysis-folium-python/