Project Report

# **GitHub URL**

https://github.com/PRAKHER0522/UCDPA\_Prakher

# **Abstract**

This project is all about understanding one such data set of uber from New York City and is very component to understand the use of data analytics and visualization. It is generated with the help of ‘PYTHON’ programming language using libraries such as NumPy,Pandas, Seaborn and MatPlotlib .Through projects like this, we can gain knowledge of various complex operations performed in data visualization. It will enable us to recognize the patterns in data of this huge organization and provides critical insights of untapped information. Also guide us in understanding the operations of MatPlotlib and Seaborn library and Machine learning concept.

# Introduction

To grow business with this competitive environment data analysis is necessary.Data analysis reports, and other kinds of analysis and report documents must be developed by businesses so that they can have references for peculiar activities and undertakings especially when making decisions for the future operations of the company. Our objective is to analyze the Uber Pickups in New York City dataset. This is more of a data visualization project that will guide us towards understanding the data and developing an intuition to understand the customers who avail the trips. Analytics is a tremendously growing niche that people apply in their businesses to give it a boost. This project will enhance our knowledge towards using the various python library for understanding the data and for developing an intuition for understanding the customers who avail the trips.

Our main objectives are:

* Visualize Uber's growth
* Characterize the demand based on identified patterns
* Estimate the value of the NYC market for Uber
* Other insights about the usage of the service
* Attempt to predict the demand's growth

# Real-world scenario

# Dataset

# Uber Pickups in New York City

For this analysis, we will be focusing on Uber pickups throughout New York City for the months of April through September 2014. The data is available across 6 CSV files, each file containing data for 1 month. It has over 500k pickups (rows) and the following 4 columns:

* Date/Time: The Date and Time of the Uber pickup
* Lat: The latitude of the pickup location
* Lon: The longitude of the pickup location
* Base: The TLC base company code affiliated with the Uber pickup

**Source :**

<https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city>

**Justification :**

From this Uber data analysis project, I need to conclude how time and place effect customer trips.

# Implementation Process

1. Importing the Essential Packages

* PANDAS
* NUMPY
* MATPLOTLIB
* SEABORN
* RE
* SKLEARN

1. Reading the Data into their designated variables like
2. Converting dataframe column into list
3. Using regexp to make column value in lower case
4. Using new column into datafarme

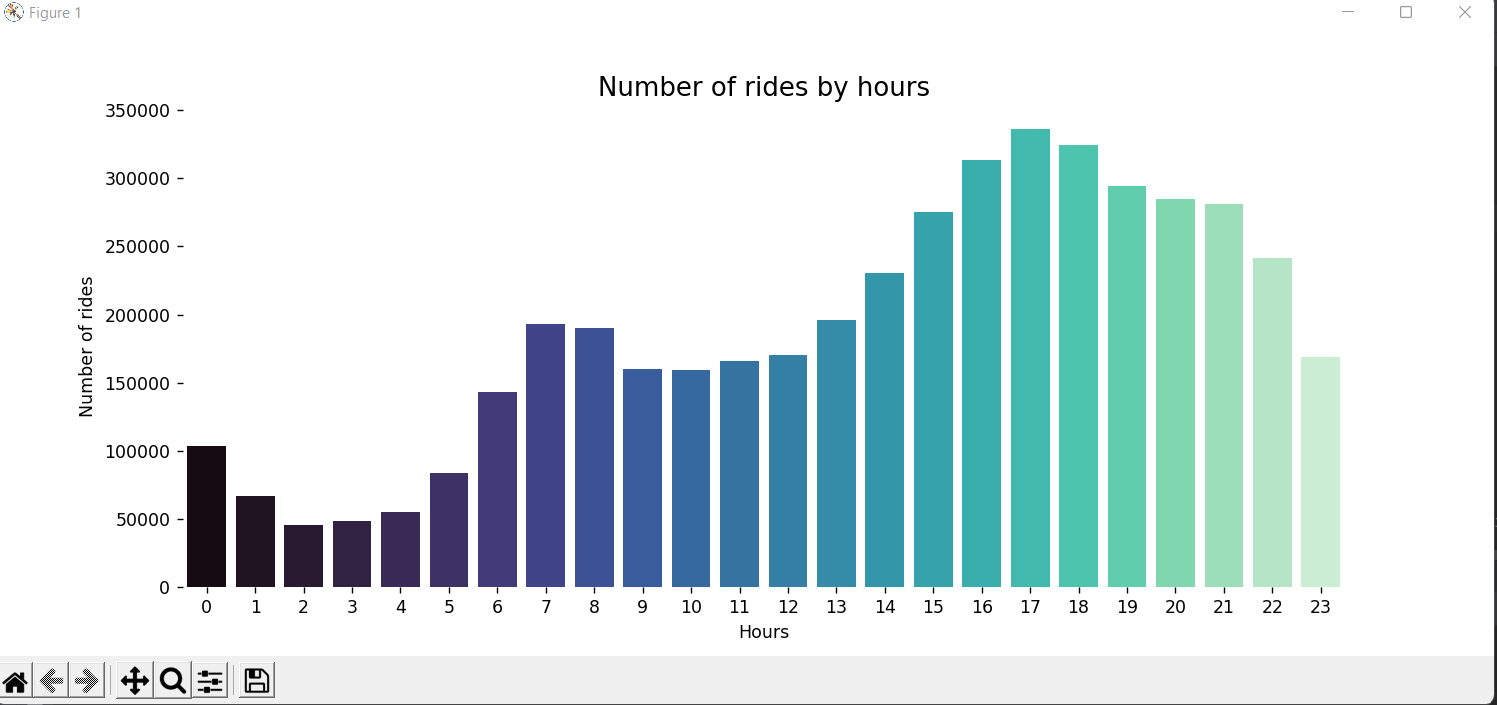
3 Collecting insights from all visualizations .Results given below

# insights

1. **Plotting the trips by the hours in a day:**

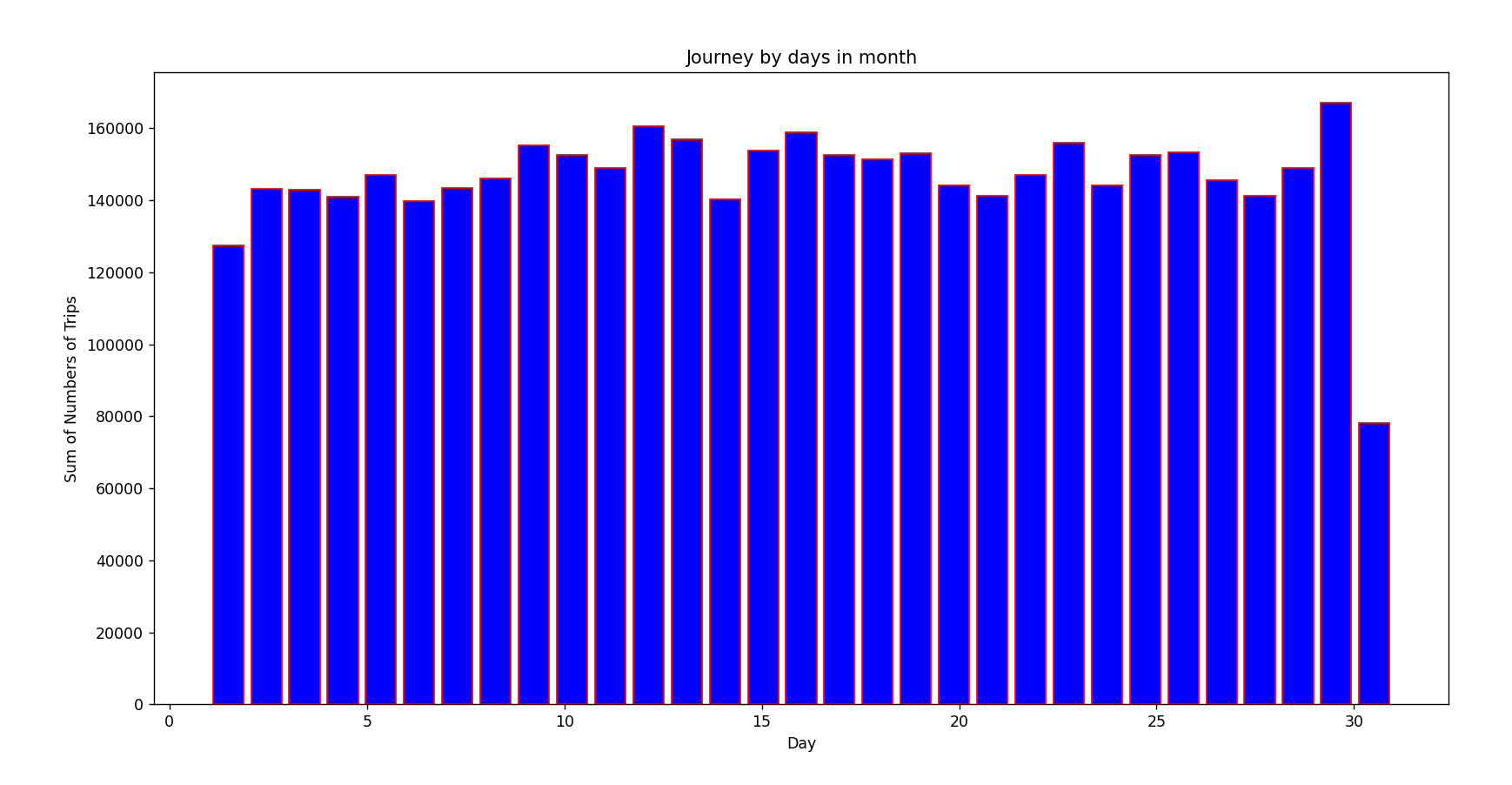
print('Plotting the trips by the hours in a day')  
plt.figure(figsize=(12,5))  
# creating bar plot to show the count of rides by hours of day  
sns.countplot(x='Hour',data=Data,palette='rocket\_r',saturation=1)  
  
  
# removing the frame around graph  
sns.despine(bottom=True, left=True)  
# removing x and y label  
plt.xlabel('Hours')  
plt.ylabel('Number of rides')  
plt.title('Number of rides by hours', fontsize=15);  
print(plt.show())

I am going to count down the total number of rides/pickups during particular hours of day for the six months (Apr - Sep). This would help us to know the busy hours of the days.



1. **Plotting data by trips during every day of the Month**

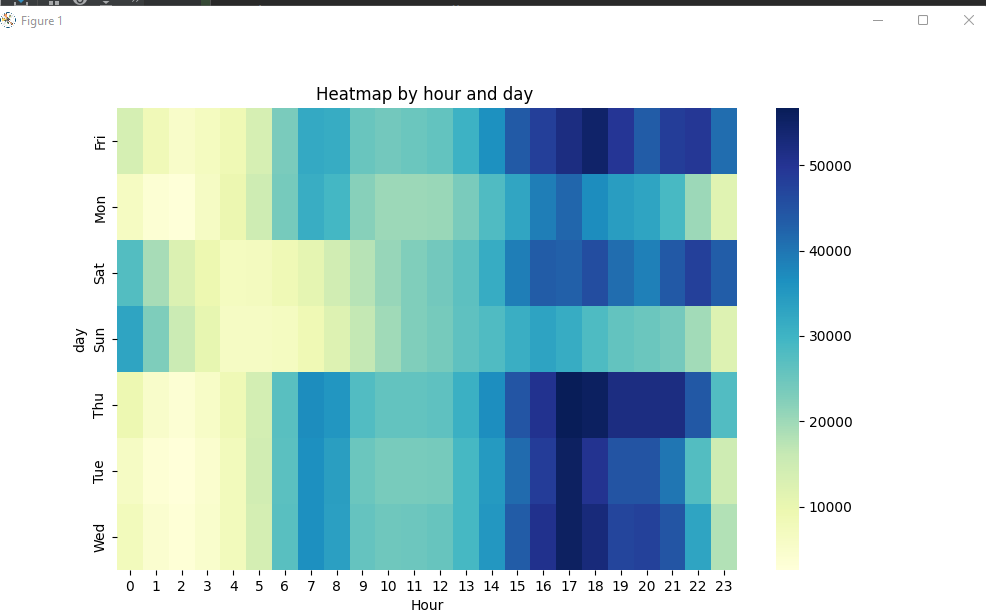
I am going to count down the total number of rides/pickups during particular day for the six months (Apr - Sep). This would help us to know the busy day of the month.



1. **Heatmap**
   1. **Heatmap by Hour and Day.**

Using Python custom function

def cust\_heat(col1,col2,col3):  
 merge\_col = Data.groupby([col1, col2]).apply(lambda x: len(x))  
 pivot = merge\_col.unstack()  
 plt.figure(figsize=(10, 6))  
 plt.title(col3)  
 return sns.heatmap(pivot, annot=False,cmap="YlGnBu")

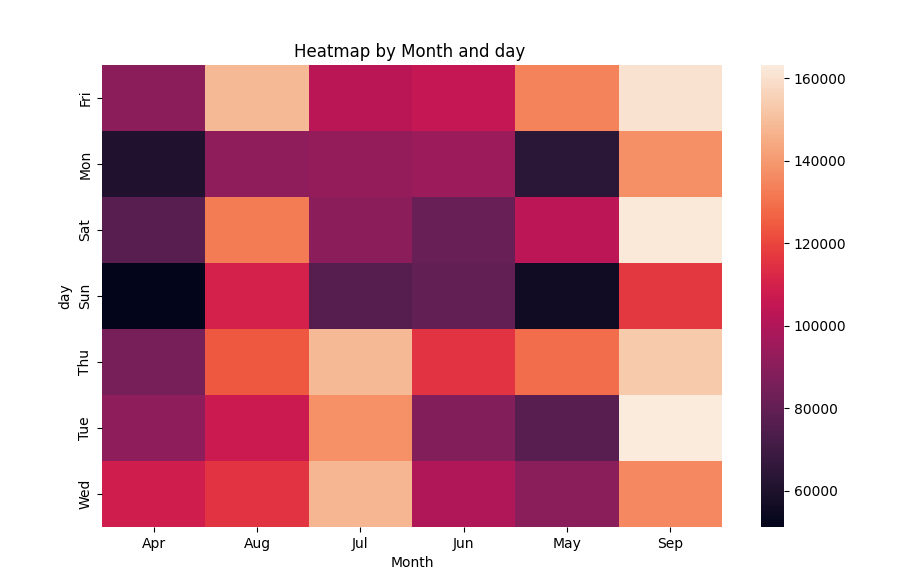


cust\_heat('day','Hour','Heatmap by hour and day ')

The heatmap indicates that morning and afternoon hours are the most busiest as expected. People often use Uber as an alternative to public transportation while going/getting out of work.

* 1. **Heatmap by Month and Day.**

def cust\_heat(col1,col2,col3):  
 merge\_col = Data.groupby([col1, col2]).apply(lambda x: len(x))  
 pivot = merge\_col.unstack()  
 plt.figure(figsize=(10, 6))  
 plt.title(col3)  
 return sns.heatmap(pivot, annot=False)

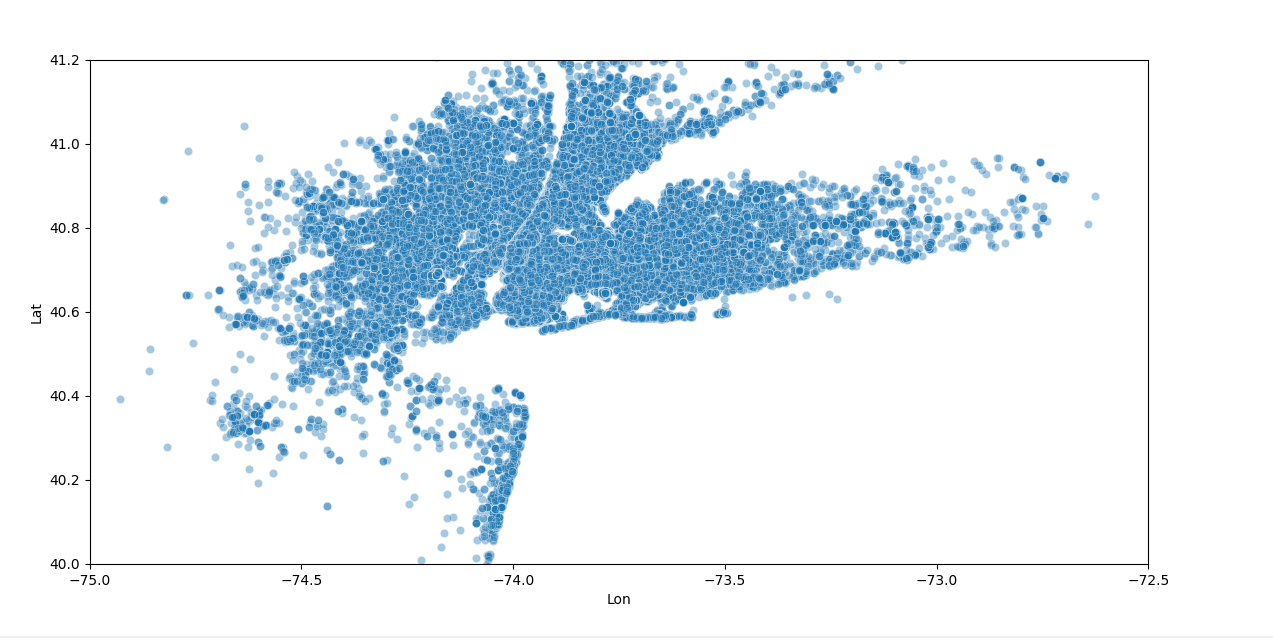


cust\_heat('day','Month','Heatmap by Month and day ')

The heatmap indicates that September month and friday are the most busiest as expected.

### 4. **Analysis of location data points**

##Analysis of location data points  
  
plt.figure(figsize=(20,10))  
sns.scatterplot(data=Data,y='Lat',x='Lon',alpha=0.4)  
plt.xlim(-75,-72.5)  
plt.ylim(40.0,41.2)  
print(plt.show())



We can see a number of hot spots here. Midtown Manhattan is clearly a huge bright spot & these are made from Midtown to Lower Manhattan.

Followed by Upper Manhattan and the Heights of Brooklyn.

# **Machine learning**

We’ll implement the DBSCAN clustering method from scikit-learn instead of structured spatial binning.The k-means algorithm is likely the most common clustering algorithm. But for spatial data, the DBSCAN algorithm is far superior in this kind of scenario where you are also using geographical data.

The DBSCAN algorithm will group points together that meet a specified density metric. Basically, we’ll define a maximum distance to make two individual points count as neighbors, as well as a minimum number of neighbors for a group of points to qualify as a cluster. The algorithm will sort the points into groups which meet the criteria and discard all of the outliers.

First, we’ll write a function which runs the clustering algorithm and returns the “hot spots.” We’ll get the coordinates of the centroid and the number of pickups in each cluster.

def get\_hot\_spots(max\_distance, min\_cars, ride\_data):  
 ## get coordinates from ride data  
 coords = ride\_data[['Lat', 'Lon']].to\_numpy()  
  
 ## calculate epsilon parameter using  
 kms\_per\_radian = 6371.0088  
 epsilon = max\_distance / kms\_per\_radian  
  
 ## perform clustering  
 db = DBSCAN(eps=epsilon, min\_samples=min\_cars,  
 algorithm='ball\_tree', metric='haversine').fit(np.radians(coords))  
  
 ## group the clusters  
 cluster\_labels = db.labels\_  
 num\_clusters = len(set(cluster\_labels))  
 clusters = pd.Series([coords[cluster\_labels == n] for n in range(num\_clusters)])  
  
 ## report  
 print('Number of clusters: {}'.format(num\_clusters))  
  
 ## initialize lists for hot spots  
 lat = []  
 lon = []  
 num\_members = []  
  
 ## loop through clusters and get centroids, number of members  
 for ii in range(len(clusters)):  
 ## filter empty clusters  
 if clusters[ii].any():  
 ## get centroid and magnitude of cluster  
 lat.append(MultiPoint(clusters[ii]).centroid.x)  
 lon.append(MultiPoint(clusters[ii]).centroid.y)  
 num\_members.append(len(clusters[ii]))  
  
 hot\_spots = [lon, lat, num\_members]  
  
 return hot\_spots

# Testing Condition :

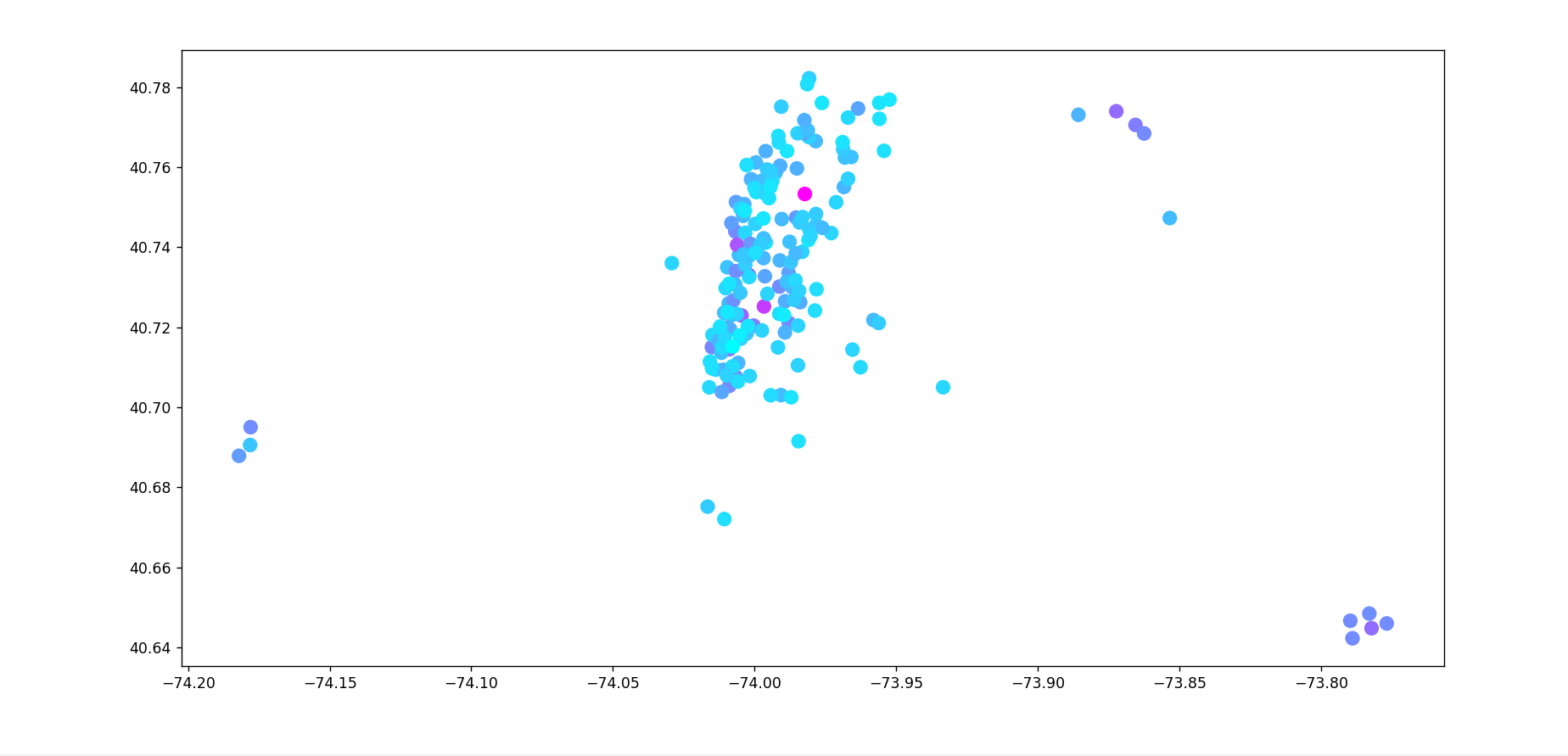
#get ride\_data  
ride\_data=Data.loc[(Data['Nday']==21) & (Data['Hour'] >15)]  
max\_distance =0.05  
min\_pickups = 25  
hot\_spots = get\_hot\_spots(max\_distance, min\_pickups, ride\_data)  
print(hot\_spots)

# Visualization :

## make the figure  
fig = plt.figure(figsize=(14, 8))  
ax = fig.add\_subplot(111)  
## set the color scale  
color\_scale = np.log(hot\_spots[2])  
# color\_scale = hot\_spots[2]  
## make the scatter plot  
plt.scatter(hot\_spots[0], hot\_spots[1], s=80, c=color\_scale, cmap=cm.cool)  
print(plt.show())

Result:

From the data, there are 172 cluster all around NYC meeting the criteria.



The map above shows areas experiencing more than 25 pickups that occur within 50 meters of each other after 4:00 PM on August 14, 2014.

# **Conclusion:**

Using a dataset with geospatial data enabled me to learn more on how to use them even better to gather insights since I don’t get to explore such kind of data. The DBSCAN algorithm is a really powerful clustering algorithm if you are dealing with geographical data

# References

https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city