Uber Pickups in New York City

Introduction

Uber Dataset is a trip data for over 20 million Uber (and other for-hire vehicles) trips in NYC. The dataset is uploaded by FiveThirtyEight on the Kaggle platform. This directory contains data on over 4.5 million Uber pickups in New York City from April to September 2014, and 14.3 million more Uber pickups from January to June 2015. Trip-level data on 10 other for-hire vehicles (FHV) companies, as well as aggregated data for 329 FHV companies, is also included. All the files are as they were received on August 3, Sept. 15, and Sept. 22, 2015.

FiveThirtyEight obtained the data from the NYC Taxi & Limousine Commission (TLC) by submitting a Freedom of Information Law request on July 20, 2015. The TLC has sent the data in batches as it continues to review trip data Uber and other HFV companies have submitted to it.

The Data

The dataset contains, roughly, four groups of files:

- Uber trip data from 2014 (April - September), separated by month, with detailed location information
- Uber trip data from 2015 (January - June), with less fine-grained location information
- Non-Uber FHV (For-Hire Vehicle) trips. The trip information varies by company but can include the day of the trip, time of the trip, pickup location, driver's for-hire license number, and vehicle's for-hire license number.
- Aggregate ride and vehicle statistics for all FHV companies (and, occasionally, for taxi companies)

However, in this project, we will use only Uber trip data from 2014 to analyze.

Uber trip data from 2014

There are six files of raw data on Uber pickups in New York City from April to September 2014. The files are separated by month and each has the following columns:

- `Date/Time` : The date and time of the Uber pickup
- `Lat` : The latitude of the Uber pickup
- `Lon` : The longitude of the Uber pickup
- `Base` : The TLC base company code affiliated with the Uber pickup

These files are named:

- `uber-raw-data-apr14.csv`
- `uber-raw-data-aug14.csv`
- `uber-raw-data-jul14.csv`
- `uber-raw-data-jun14.csv`
- `uber-raw-data-may14.csv`
- `Uber-raw-data-sep14.csv`

Analysis

In this project, I have directly imported the Uber Dataset from Kaggle to Google Colab using Kaggle API without uploading it to the Google Colab platform.

I have analyzed how many rides were booked based on the TLC company base code, month, day, hour, and combinations of them in New York City. And also I have done a Geo-Spatial analysis of rides booked in NYC on Sunday.

Code

Uber Drive Data Analysis

Installing Uber Drive Dataset From Kaggle

```
[1] # installing kaggle library in Google Colab Platform
    ! pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.64.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2021.10.8)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (6.1.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (2.10)
```

```
[2] # creating a directory named kaggle
    ! mkdir ~/.kaggle
```

```
[3] # copying essential kaggle.json file from MyDrive to kaggle directory
    ! cp /content/drive/MyDrive/kaggle.json ~/.kaggle/kaggle.json
```

```
# downloading Uber Data from kaggle
! kaggle datasets download fivethirtyeight/uber-pickups-in-new-york-city
```

```
Downloading uber-pickups-in-new-york-city.zip to /content
 90% 98.0M/109M [00:01<00:00, 78.1MB/s]
100% 109M/109M [00:01<00:00, 74.0MB/s]
```

[5]
```
# unzipping the zipped files
! unzip uber-pickups-in-new-york-city
```

```
Archive:  uber-pickups-in-new-york-city.zip
  inflating: Uber-Jan-Feb-FOIL.csv
  inflating: other-American_B01362.csv
  inflating: other-Carmel_B00256.csv
  inflating: other-Dial7_B00887.csv
  inflating: other-Diplo_B01196.csv
  inflating: other-FHV-services_jan-aug-2015.csv
  inflating: other-Federal_02216.csv
  inflating: other-Firstclass_B01536.csv
  inflating: other-Highclass_B01717.csv
  inflating: other-Lyft_B02510.csv
  inflating: other-Prestige_B01338.csv
  inflating: other-Skyline_B00111.csv
  inflating: uber-raw-data-apr14.csv
  inflating: uber-raw-data-aug14.csv
  inflating: uber-raw-data-janjune-15.csv
  inflating: uber-raw-data-jul14.csv
  inflating: uber-raw-data-jun14.csv
  inflating: uber-raw-data-may14.csv
  inflating: uber-raw-data-sep14.csv
```

Importing Essential Libraries

[6]
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
```

Reading Datasets

The Uber Dataset contains 6 individual files representing its rides in each month from april to september.

Reading csv files using pandas

[7]
```
apr_data = pd.read_csv("/content/uber-raw-data-apr14.csv")
may_data = pd.read_csv("/content/uber-raw-data-may14.csv")
jun_data = pd.read_csv("/content/uber-raw-data-jun14.csv")
jul_data = pd.read_csv("/content/uber-raw-data-jul14.csv")
aug_data = pd.read_csv("/content/uber-raw-data-aug14.csv")
sep_data = pd.read_csv("/content/uber-raw-data-sep14.csv")
```

[8]
```
# concatenating datasets
df = pd.concat([apr_data, may_data, jun_data, jul_data, aug_data, sep_data], axis=0)
```

Data Preprocessing

```
[9] # shape of the dataset
df.shape
```

```
(4534327, 4)
```

Dataset consists of over 4.5 million rows and only 4 columns.

Checking for null values

```
[10] df.isnull().sum()
```

```
Date/Time    0
Lat          0
Lon          0
Base         0
dtype: int64
```

There are no null values in the dataframe

```
[11] # overview of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4534327 entries, 0 to 1028135
Data columns (total 4 columns):
 #   Column     Dtype
---  ------     -----
 0   Date/Time  object
 1   Lat        float64
 2   Lon        float64
 3   Base       object
dtypes: float64(2), object(2)
memory usage: 173.0+ MB
```

Changing the format of Date/Time column

```
[12] # converting the data type of Date/Time column into pandas datetime
df['Date/Time'] = pd.to_datetime(df['Date/Time'], format="%m/%d/%Y %H:%M:%S")
```

```
[13] # data type of each column
df.dtypes
```

```
Date/Time    datetime64[ns]
Lat                 float64
Lon                 float64
Base                 object
dtype: object
```

Splitting the Date/Time column into several columns

```
[14] df['month'] = df['Date/Time'].dt.month
df['weekday'] = df['Date/Time'].dt.day_name()
df['day'] = df['Date/Time'].dt.day
df['hour'] = df['Date/Time'].dt.hour
df['minute'] = df['Date/Time'].dt.minute
```

```
[15] # top five rows of the dataframe
df.head()
```

	Date/Time	Lat	Lon	Base	month	weekday	day	hour	minute
0	2014-04-01 00:11:00	40.7690	-73.9549	B02512	4	Tuesday	1	0	11
1	2014-04-01 00:17:00	40.7267	-74.0345	B02512	4	Tuesday	1	0	17
2	2014-04-01 00:21:00	40.7316	-73.9873	B02512	4	Tuesday	1	0	21
3	2014-04-01 00:28:00	40.7588	-73.9776	B02512	4	Tuesday	1	0	28
4	2014-04-01 00:33:00	40.7594	-73.9722	B02512	4	Tuesday	1	0	33

Exploratory Data Analysis on Uber Dataset

```
[16] # setting seaborn theme
     custom_params = {"axes.spines.right": False, "axes.spines.top": False}
     sns.set_theme(style="ticks", rc=custom_params)
     sns.set_palette("bright")
```

Number of rides booked from each base

```
[17] # number of rides booked from each Base
     df['Base'].value_counts()
```

```
B02617    1458853
B02598    1393113
B02682    1212789
B02764     263899
B02512     205673
Name: Base, dtype: int64
```

Instead of viewing the data in text format, it is preferrable to use visualization.

```
[18] # plotting the data using seaborn countplot
     sns.countplot(data = df, x ='Base')
     plt.xlabel('TLC Base Company Code') # x-axis name
     plt.ylabel('Count') # y-axis name
     plt.yticks(ticks=np.arange(100000, 1600000, 100000)) # y-axis markings
     plt.title('Number of bookings respective to the base') # title name
     plt.show()
```

B02617 base bookings was used the most while B02512 base was used the least.

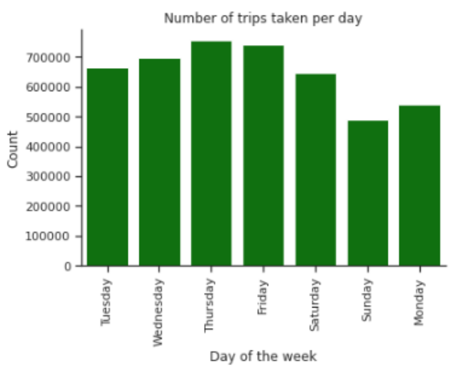
Number of trips in each month

[19]
```
# plotting the data using seaborn countplot
sns.countplot(data = df, x = df['month'], color = 'red')
plt.xlabel('Month') # x-axis name
plt.xticks(ticks=[0,1,2,3,4,5],labels=['Apr','May','Jun','Jul','Aug','Sep']) # x-axis markings
plt.ylabel('Count') # y-axis name
plt.yticks(ticks=np.arange(100000, 1100000, 100000)) # y-axis markings
plt.title('Number of trips taken in each month')
plt.show()
```

Uber rides gradually increased from April to September.

Number of trips at each day

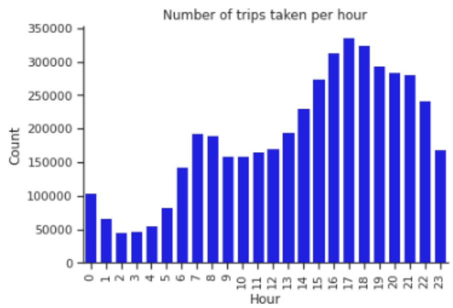
[20]
```
# plotting the data using seaborn countplot
sns.countplot(data = df, x = df['weekday'], color = 'green')
plt.xlabel('Day of the week') # x-axis name
plt.xticks(rotation=90)
plt.ylabel('Count') # y-axis name
plt.title('Number of trips taken per day')
plt.show()
```



Thursday and the weekend was the busiest days in a week for the Uber.

Number of trips at each hour

[21]
```
# plotting the data using seaborn countplot
sns.countplot(data = df, x = df['hour'], color = 'blue')
plt.xlabel('Hour') # x-axis name
plt.xticks(rotation=90) # rotating x-axis names vertically
plt.ylabel('Count') # y-axis name
plt.title('Number of trips taken per hour')
plt.show()
```



From the graph above, we can see that most of the Uber trips were taken at 05:00 pm.

Number of trips taken at each hour in each month

Grouping the dataset by **hour** and **month** columns using pandas groupby function.

[22]
```
temp_df = df.groupby(['hour','month']).size().reset_index()
temp_df
```

	hour	month	0
0	0	4	11910
1	0	5	13875
2	0	6	14514
3	0	7	17953
4	0	8	21451
...	...	...	...
139	23	5	24836
140	23	6	24182
141	23	7	29346
142	23	8	33609
143	23	9	36568

144 rows × 3 columns

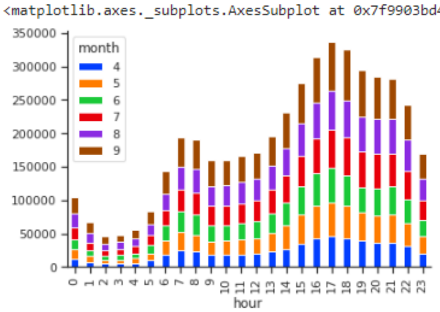
Pivoting the modified dataframe with its index as **hour** and columns as **month**.

```
[23] temp_df = temp_df.pivot_table(index='hour',columns='month',values=0)
     temp_df
```

month hour	4	5	6	7	8	9
0	11910	13875	14514	17953	21451	24133
1	7769	8186	9167	11527	14471	16107
2	4935	5372	6189	8562	10105	10702
3	5040	5946	6937	9199	10376	10789
4	6095	6945	7701	10040	11774	12675
5	9476	10789	11955	14932	16525	20262
6	18498	21015	22030	23456	24907	33307
7	24924	27413	30834	32545	34064	43314
8	22843	25460	29771	33387	34566	44477
9	17939	20507	24298	28486	30195	38542
10	17865	20801	23584	28558	30706	37634
11	18774	22055	24155	30120	31778	38821
12	19425	23595	25233	30900	32106	39193
13	22603	27699	28937	35832	35764	45042
14	27190	34363	34428	41357	40644	52643
15	35324	43087	41586	46053	48197	61219
16	42003	49127	48162	52403	53481	68224
17	45475	51508	50452	58260	57122	73373
18	43003	48965	45013	57268	55390	75040
19	38923	42387	38203	52332	53008	69660
20	36244	40731	40108	51859	51674	63988
21	36964	42217	40791	49528	51354	60606
22	30645	35556	35614	42218	46008	51817
23	20649	24836	24182	29346	33609	36568

```
[24] # using bar plot in pandas for plotting stacked bar graph
     temp_df.plot(kind='bar',stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9903bd4450>
```



Rather than the previous graph this stacked graph shows us the number of rides booked with monthly divison.

By this graph we can conclude that comparably most of the rides are booked during August and September months during office leaving hours.

Number of trips per month on different bases

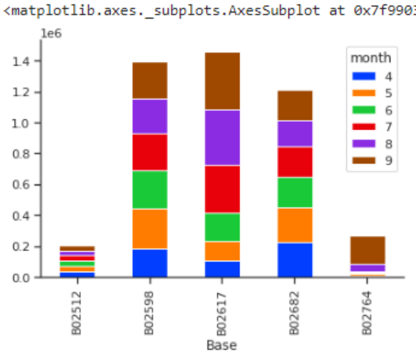
Grouping the dataset by **Base** and **month** columns and then *pivoting* the dataset with its index as **Base** and columns as **month**.

```
[25] temp_df = df.groupby(['Base','month']).size().reset_index().pivot_table(index='Base',columns='month',values=0)
     temp_df
```

month Base	4	5	6	7	8	9
B02512	35536	36765	32509	35021	31472	34370
B02598	183263	260549	242975	245597	220129	240600
B02617	108001	122734	184460	310160	355803	377695
B02682	227808	222883	194926	196754	173280	197138
B02764	9908	9504	8974	8589	48591	178333

```
[26] # using bar plot in pandas
     temp_df.plot(kind='bar',stacked=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f990395f550>



As previously noticed the bases B02598, B02617, B02682 were booked the most.

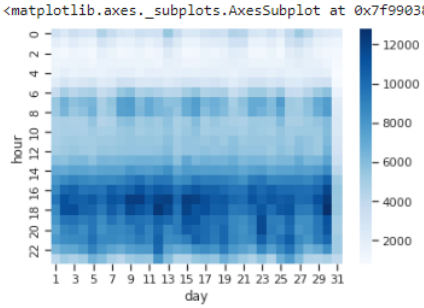
Heatmaps

Heatmap of trips on each day at each hour

Grouping the dataset by **day** and **hour** columns and then *pivoting* the dataset with its index as **day** and columns as **hour**.

```
[27] temp_df = df.groupby(['hour','day']).size().reset_index().pivot_table(index='hour',columns='day',values=0)
     # heatmap function to visualize
     sns.heatmap(temp_df, cmap='Blues')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f99038c8b10>



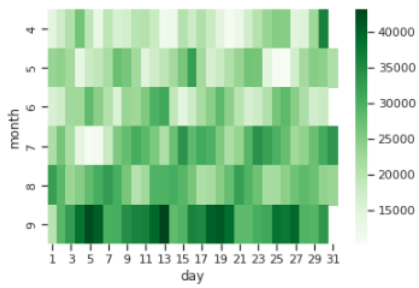
Brighter blue color means most rides and dark blue indicates less booking of Uber rides. Based on the heatmap we can conclude that most of the rides were taken in between 4:00 pm to 10:00 pm.

Heatmap of trips in each month during each day

Grouping the dataset by **month** and **day** columns and then *pivoting* the dataset with its index as **month** and columns as **day**.

```
[28] temp_df = df.groupby(['month','day']).size().reset_index().pivot_table(index='month',columns='day',values=0)
     # heatmap function to visualize
     sns.heatmap(temp_df,cmap='Greens')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f99038099d0>
```



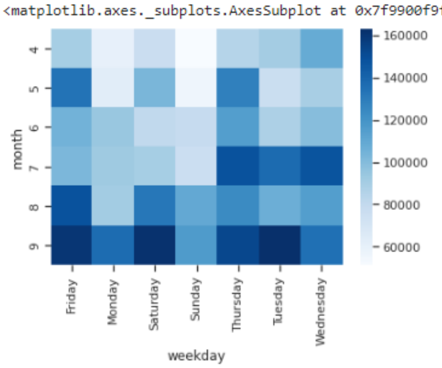
On April 30th, Sep 5th, 13th, 19th, 27th most of the rides were booked.

Heatmap of trips in each month based on weekday

Grouping the dataset by **month** and **weekday** columns and then *pivoting* the dataset with its index as **month** and columns as **weekday**.

```
[29] temp_df = df.groupby(['month','weekday']).size().reset_index().pivot_table(index='month',columns='weekday',values=0)
     # heatmap function to visualize
     sns.heatmap(temp_df,cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9900f9fc90>
```



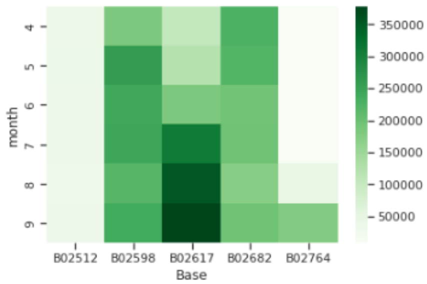
Almost in every month most of the rides were booked on Thursday, Friday and Saturday.

Heatmap of trips during every month based of Base they were booked

Grouping the dataset by **month** and **Base** columns and then *pivoting* the dataset with its index as **month** and columns as **Base**.

```
[30] temp_df = df.groupby(['month','Base']).size().reset_index().pivot_table(index='month',columns='Base',values=0)
     # heatmap function to visualize
     sns.heatmap(temp_df,cmap='Greens')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9900f9f050>
```



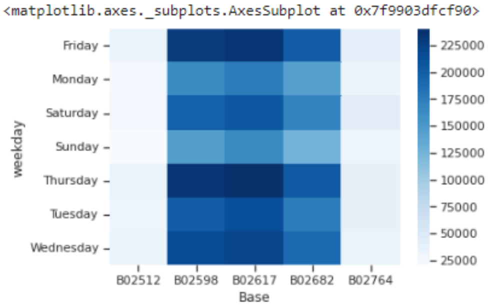
High rides were booked on B02617 Base during August and September months.

Heatmap of trips on each day based on the Base they were booked

Grouping the dataset by **weekday** and **Base** columns and then *pivoting* the dataset with its index as **weekday** and columns as **Base**.

```
[31] temp_df = df.groupby(['weekday','Base']).size().reset_index().pivot_table(index='weekday',columns='Base',values=0)
     # heatmap function to visualize
     sns.heatmap(temp_df,cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9903dfcf90>
```



From this heatmap, as we previously noticed through countplot, most of the uber rides were booked on Thursday and Friday from B02617 and B02598 bases. And from bases B02512 and B02764 harldy rides were booked.

Geo Spatial Analysis

Initialising the map

[32]
```
# Map in folium
basemap = folium.Map(width=500,height=500, zoom_start=4)
```

```
basemap
```

Mapping of the Uber rides taken on sunday of every week in every month

[34]
```
# extracting the rides which were taken on Sunday
temp_df = df[df['weekday']=='Sunday']
```

[35]
```
# top 5 rows
temp_df.head()
```

	Date/Time	Lat	Lon	Base	month	weekday	day	hour	minute
6965	2014-04-06 00:00:00	40.6547	-74.3033	B02512	4	Sunday	6	0	0
6966	2014-04-06 00:00:00	40.7356	-74.0006	B02512	4	Sunday	6	0	0
6967	2014-04-06 00:00:00	40.7421	-74.0041	B02512	4	Sunday	6	0	0
6968	2014-04-06 00:00:00	40.7401	-74.0053	B02512	4	Sunday	6	0	0
6969	2014-04-06 00:01:00	40.7368	-73.9877	B02512	4	Sunday	6	0	1

The dataframe is grouped by latitude and longitude and the values as count.

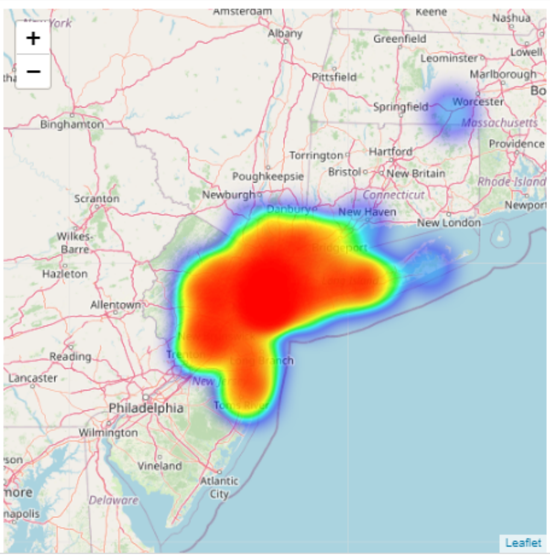
```
[36] # Grouping the dataframe by latitude and longitude
     temp_df = temp_df.groupby(['Lat','Lon']).size().reset_index()
     temp_df
```

	Lat	Lon	0
0	39.9374	-74.0722	1
1	39.9378	-74.0721	1
2	39.9384	-74.0742	1
3	39.9385	-74.0734	1
4	39.9415	-74.0736	1
...	...	...	...
209225	41.3141	-74.1249	1
209226	41.3180	-74.1298	1
209227	41.3195	-73.6905	1
209228	41.3197	-73.6903	1
209229	42.1166	-72.0666	1

209230 rows × 3 columns

We will use Heatmap from folium.plugins to map the uber rides. The important parameter of the function is location which should be a numpy array.

```
[38] # Adding the Heatmap to our basemap
     HeatMap(ls).add_to(basemap)
     # visualizing the map
     basemap
```



The map shows the Heatmap of Uber rides which were taken on Sunday in and around New York.

Conclusion

Based on the analysis done above, we can conclude the following points.

1. Only 3 out of 5 bases were used the most.
2. The Uber rides gradually increased with time (from April to September months).
3. Uber rides were booked the most during office leaving hours.
4. The Uber services were widely used in between 07:00 AM to 10:00 PM each day.
5. Thursday, Friday and Saturday were the busiest days of Uber.
6. Uber rides were booked not only in New-York city alone but also outside of NYC (Sturbridge and Bridgehampton).

Data Source

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

References

➔ https://thecleverprogrammer.com/2021/04/21/uber-trips-analysis-using-python/

➔ https://www.analyticsvidhya.com/blog/2021/10/end-to-end-predictive-analysis-on-ubers-data/

➔ https://www.youtube.com/watch?v=EWp1LM9KVRs&list=PLfFghEzKVmjuUzJtZkI38zqMEKb1yCH91&index=2