ITCS-6100 BIG DATA FOR COMPUTATIONAL ADVANTAGE
GROUP-13 PROJECT FINAL DOCUMENTATION: New York Taxi Data Analysis and Patterns

1. A. TEAM MEMBERS:

- Vineeth Avula
- Srikar Gaddipati
- Likhitha Alla
- Nikhita Somanchi
- Kamala Kumari Karuturi

1.B. COMMUNICATION PLAN:

- Team members will discuss perspectives through slack and exchange their ideas apparently whenever it's required.
- To monitor and get the required results, all the team members will gather via Zoom or Google meet and will finish the tasks accordingly.
- The project's repository can be accessed on GitHub using URL that's given below: https://github.com/AvulaVineeth/Group13

2. BUSINESS PROBLEM OR OPPORTUNITY:

There are a lot of people who use the services provided by an American cab mobility company on a regular basis whose headquarters is in New York. Services were offered through a mobile application where it can associate its customers with nearby drivers whoever is available. The sturdy nature of our mobile application will allow us to cope with substantial loads whenever we want. Even if the organization is doing well, it is not to meet the sudden raise in the demand at some time even though it has the full capacity to function. The organization decides to make some money by raising the charge based on the ongoing demand. In case there is no demand in the future it would be very difficult to implement.

3. SELECTION OF DATA:

The dataset that's chosen (New York City Taxi and Limousine Commission (TLC) Trip Record Data) is taken from a URL that's related to the NYC government. TLC is the one who published it. The one that is in charge of giving permits and enforcing regulations over Medallion(Yellow) taxi cabs, in case of for-hire vehicles (including commuter vans, black cars, and opulent limousines), community-based liveries, and paratransit vehicles in New York City Taxi and

Limousine Commission (TLC) which was founded in 1971.

- **LICENSE:** https://www1.nyc.gov/home/terms-of-use.page
- **DOCUMENTATION:** http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
- **DATASET:** https://registry.opendata.aws/nyc-tlc-trip-records-pds/

4. PREDICTION ANALYSIS:

Our objective is to delve into the dataset and work on the findings to figure out the optimal solution for the company to rely on, in reaching their demand forecast goals. We believe the research outcomes will provide insights in greater depth into their requirement and enable them to achieve their business objective.

- ✓ **Pick-up hourly distribution:**

  On examining the given dataset, we thoroughly analyze and predict the number of trips on an hourly basis.

- ✓ **Pick-up weekday distribution:**

  In addition to this, we also perform analysis on how well the cabs and taxis are performing on a day-to-day basis. It generates a report of the rides booked on each day, through this we can assess which days of the week are the most and least busy.

- ✓ **Vendor pick-up hour density, by weekday:**

  In addition to this, we also determine the vendor pickup density every single day with a detailed hourly description as well. Through this, a demonstration of busy hours in a day for vendor pick-ups can be examined.

5.RESEARCH OBJECTIVES AND QUESTIONS:

Our plan is to search in the dataset and find a solution to the existing problem that is being faced in the organization in such a way that there will not be any problems in the future. The main aim is to provide a follow-up based on the demand and assist in reaching the business goals. To carry out the solution we decided to use AWS technologies. We are planning to execute relevant designs and algorithms to find out the flow in demand. The main task now is to find out the relevant design and algorithms that are most appropriate in examining the dataset.

6. DATA UNDERSTANDING:

By understanding the nature of the data, we can predict that:

- ✓ As per the dataset obtained, it consists of 13,69,765 rows and 19 columns in total.
  - PULocationID – It consists of the pickup locations of the customers.
  - DOLocationID – It consists of the drop-off locations.

- Passenger Count- It consists of Passenger count in a ride.
- Trip distance – It consists of the total distance of a trip.

✓ Alongside there are several other columns such as RatecodeID, store_and_fwd_flag,vendorID, payment type, fare_amount, taxes, tolls, and some other columns.

In [3]:
```
#looking through the data
data.head()
```

Out[3]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payme
0	1.0	2021-01-01 00:30:10	2021-01-01 00:36:12	1.0	2.10	1.0	N	142	43	
1	1.0	2021-01-01 00:51:20	2021-01-01 00:52:19	1.0	0.20	1.0	N	238	151	
2	1.0	2021-01-01 00:43:30	2021-01-01 01:11:06	1.0	14.70	1.0	N	132	165	
3	1.0	2021-01-01 00:15:48	2021-01-01 00:31:01	0.0	10.60	1.0	N	138	132	
4	2.0	2021-01-01 00:31:49	2021-01-01 00:48:21	1.0	4.94	1.0	N	68	33	

In [4]:
```
#describing the statistical information of the data
data.describe()
```

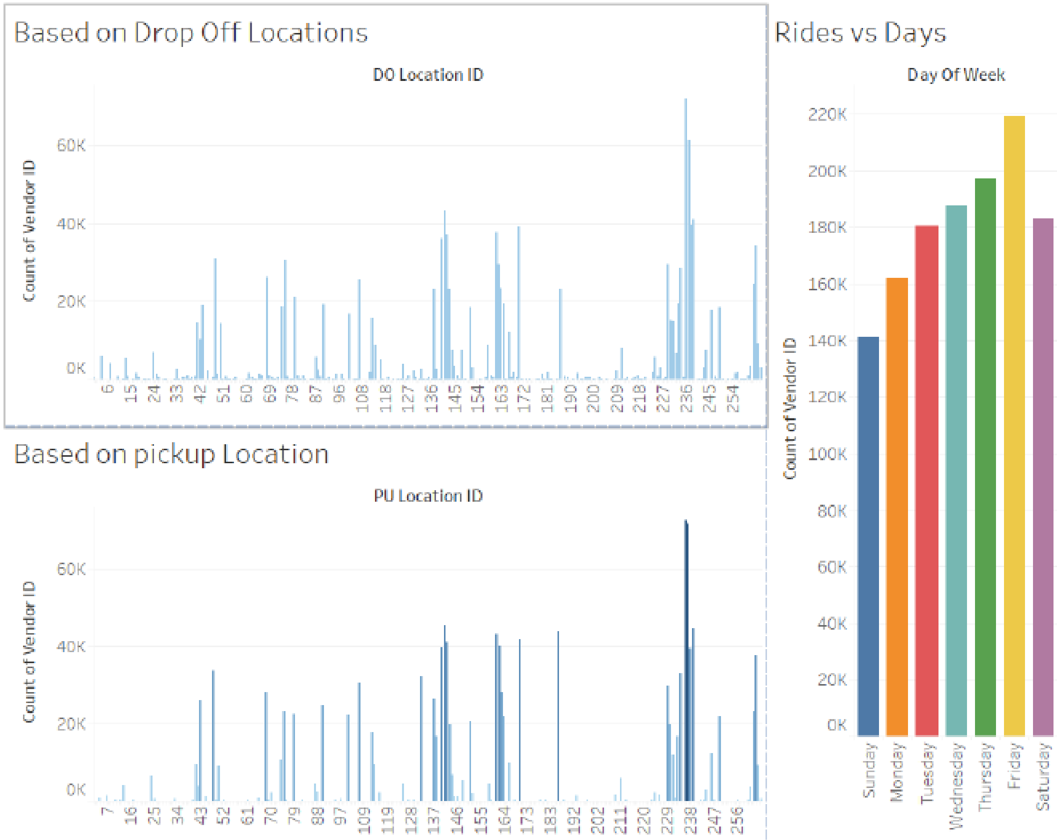
Out[4]:

	VendorID	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	
count	1.271413e+06	1.271413e+06	1.369765e+06	1.271413e+06	1.369765e+06	1.369765e+06	1.271413e+06	1.369765e+06	1.369765e+06	1.369765e+06	1
mean	1.676925e+00	1.411508e+00	4.631982e+00	1.035081e+00	1.652472e+02	1.614956e+02	1.280521e+00	1.209662e+01	9.705085e-01	4.930411e-01	1
std	4.676513e-01	1.059833e+00	3.939042e+02	5.994840e-01	6.783849e+01	7.210800e+01	4.916921e-01	1.291338e+01	1.231256e+00	7.632070e-02	2
min	1.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	-4.900000e+02	-5.500000e+00	-5.000000e-01	-1
25%	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.240000e+02	1.070000e+02	1.000000e+00	6.000000e+00	0.000000e+00	5.000000e-01	0
50%	2.000000e+00	1.000000e+00	1.700000e+00	1.000000e+00	1.620000e+02	1.620000e+02	1.000000e+00	8.500000e+00	0.000000e+00	5.000000e-01	1
75%	2.000000e+00	1.000000e+00	3.020000e+00	1.000000e+00	2.360000e+02	2.360000e+02	2.000000e+00	1.350000e+01	2.500000e+00	5.000000e-01	2
max	2.000000e+00	8.000000e+00	2.631633e+05	9.900000e+01	2.650000e+02	2.650000e+02	4.000000e+00	6.960500e+03	8.250000e+00	5.000000e-01	1

```
In [5]: #getting info of all the columns
        data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1369765 entries, 0 to 1369764
Data columns (total 19 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   VendorID               1271413 non-null  float64
 1   tpep_pickup_datetime   1369765 non-null  object
 2   tpep_dropoff_datetime  1369765 non-null  object
 3   passenger_count        1271413 non-null  float64
 4   trip_distance          1369765 non-null  float64
 5   RatecodeID             1271413 non-null  float64
 6   store_and_fwd_flag     1271413 non-null  object
 7   PULocationID           1369765 non-null  int64
 8   DOLocationID           1369765 non-null  int64
 9   payment_type           1271413 non-null  float64
 10  fare_amount            1369765 non-null  float64
 11  extra                  1369765 non-null  float64
 12  mta_tax                1369765 non-null  float64
 13  tip_amount             1369765 non-null  float64
 14  tolls_amount           1369765 non-null  float64
 15  improvement_surcharge  1369765 non-null  float64
 16  total_amount           1369765 non-null  float64
 17  congestion_surcharge   1369765 non-null  float64
 18  ID                     1369765 non-null  int64
dtypes: float64(13), int64(3), object(3)
memory usage: 198.6+ MB

In [6]: #checking number of rows and columns
        data.shape

Out[6]: (1369765, 19)
```

EXPLARATORY DATA ANALYSIS:

- ✓ From the datasets, we can analyse that there are 258 unique pickup locations and 260 drop locations.
- ✓ On understanding, we can explain that Friday has the highest number of rides, Next comes Thursday and Wednesday, and the least number of rides are on Sunday.
- ✓ Amongst the rides, the most number of passenger count for rides is 1 and followed by 2.
- ✓ Amongst the drop-off locations, these are the descending order of locations with the highest number of drop-offs 236,237,239,238,141.
- ✓ Amongst the pick-up locations, these are in descending order of locations with the highest number of pick-ups 236, and 237,141,239,186.



7. DATA PREPARATION:

In this stage, we prepare our data for analysis. Here, we employ several data-cleaning techniques.

Data Cleaning

Checking how many rows with atleast one column having Null value

In [7]:
```
data.isnull().sum() * 100 / len(data)
```

Out[7]:
```
VendorID                 7.18021
tpep_pickup_datetime     0.00000
tpep_dropoff_datetime    0.00000
passenger_count          7.18021
trip_distance            0.00000
RatecodeID               7.18021
store_and_fwd_flag       7.18021
PULocationID             0.00000
DOLocationID             0.00000
payment_type             7.18021
fare_amount              0.00000
extra                    0.00000
mta_tax                  0.00000
tip_amount               0.00000
tolls_amount             0.00000
improvement_surcharge    0.00000
total_amount             0.00000
congestion_surcharge     0.00000
ID                       0.00000
dtype: float64
```

Our data has around 7.1% rows with null values in columns 'RatecodeID', 'store_and_fwd_flag', 'congestion_surcharge', 'passenger_count'. As the amount of rows with Null values are less as compared to total rows we have, we will drop all rows having atleast one column with Null value.

Imputation of the rows containing null value

In [8]:
```
# Performing imputation with mode
data['VendorID'] = data['VendorID'].fillna(data['VendorID'].mode()[0])
data['payment_type'] = data['payment_type'].fillna(data['payment_type'].mode()[0])
data['store_and_fwd_flag'] = data['store_and_fwd_flag'].fillna(data['store_and_fwd_flag'].mode()[0])
data['passenger_count'] = data['passenger_count'].fillna(data['passenger_count'].mode()[0])
data['RatecodeID'] = data['RatecodeID'].fillna(data['RatecodeID'].mode()[0])

data.isnull().sum() * 100 / len(data)
```

Out[8]:
```
VendorID                 0.0
tpep_pickup_datetime     0.0
tpep_dropoff_datetime    0.0
passenger_count          0.0
trip_distance            0.0
RatecodeID               0.0
store_and_fwd_flag       0.0
PULocationID             0.0
DOLocationID             0.0
payment_type             0.0
fare_amount              0.0
extra                    0.0
mta_tax                  0.0
tip_amount               0.0
tolls_amount             0.0
improvement_surcharge    0.0
total_amount             0.0
congestion_surcharge     0.0
ID                       0.0
dtype: float64
```

Correcting datatype of tpep_pickup_datetime and tpep_dropoff_datetime column

In [9]:
```
data['tpep_pickup_datetime'] = pd.to_datetime(data['tpep_pickup_datetime'])
data['tpep_dropoff_datetime'] = pd.to_datetime(data['tpep_dropoff_datetime'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1271413 entries, 0 to 1271412
Data columns (total 19 columns):
VendorID                 1271413 non-null float64
tpep_pickup_datetime     1271413 non-null datetime64[ns]
tpep_dropoff_datetime    1271413 non-null datetime64[ns]
passenger_count          1271413 non-null float64
trip_distance            1271413 non-null float64
RatecodeID               1271413 non-null float64
store_and_fwd_flag       1271413 non-null object
PULocationID             1271413 non-null int64
DOLocationID             1271413 non-null int64
payment_type             1271413 non-null float64
fare_amount              1271413 non-null float64
extra                    1271413 non-null float64
mta_tax                  1271413 non-null float64
tip_amount               1271413 non-null float64
tolls_amount             1271413 non-null float64
improvement_surcharge    1271413 non-null float64
total_amount             1271413 non-null float64
congestion_surcharge     1271413 non-null float64
ID                       1271413 non-null int64
dtypes: datetime64[ns](2), float64(13), int64(3), object(1)
memory usage: 194.0+ MB
```

Checking anomalies with Pickup time and drop time

Pickup time will be always before drop-off time. We need to find rows with anomalies with trip duration.

In [10]:
```python
pd_time_anomaly = len(data[data['tpep_pickup_datetime'] >= data['tpep_dropoff_datetime']])
print(f"Total rows with anomalies with trip duration: {pd_time_anomaly}")
```

```
Total rows with anomalies with trip duration: 794
```

In [11]:
```python
# Removing all rows anomalies with trip duration
data = data[data['tpep_pickup_datetime'] < data['tpep_dropoff_datetime']]
pd_time_anomaly = len(data[data['tpep_pickup_datetime'] > data['tpep_dropoff_datetime'] ])
print(f"Now, total rows with anomalies with trip duration: {pd_time_anomaly}")
```

```
Now, total rows with anomalies with trip duration: 0
```

Changing categorical variable into numeric/boolean ¶

All input and output variables in machine learning models must be numeric. This means that if our data contains categorical data, we must convert it to numbers before fitting and evaluating a model. Here, 'store_and_fwd_flag' is Y/N. So, we will convert it to boolean values 0 or 1.

In [24]:
```python
data['store_and_fwd_flag'] = data['store_and_fwd_flag'].replace({'N': 0, 'Y': 1})
data.head()
```

Out[24]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payme
0	1.0	2021-01-01 00:30:10	2021-01-01 00:36:12	1.0	2.10	1.0	0	142	43	
1	1.0	2021-01-01 00:51:20	2021-01-01 00:52:19	1.0	0.20	1.0	0	238	151	
2	1.0	2021-01-01 00:43:30	2021-01-01 01:11:06	1.0	14.70	1.0	0	132	165	
4	2.0	2021-01-01 00:31:49	2021-01-01 00:48:21	1.0	4.94	1.0	0	68	33	
5	1.0	2021-01-01 00:16:29	2021-01-01 00:24:30	1.0	1.60	1.0	0	224	68	

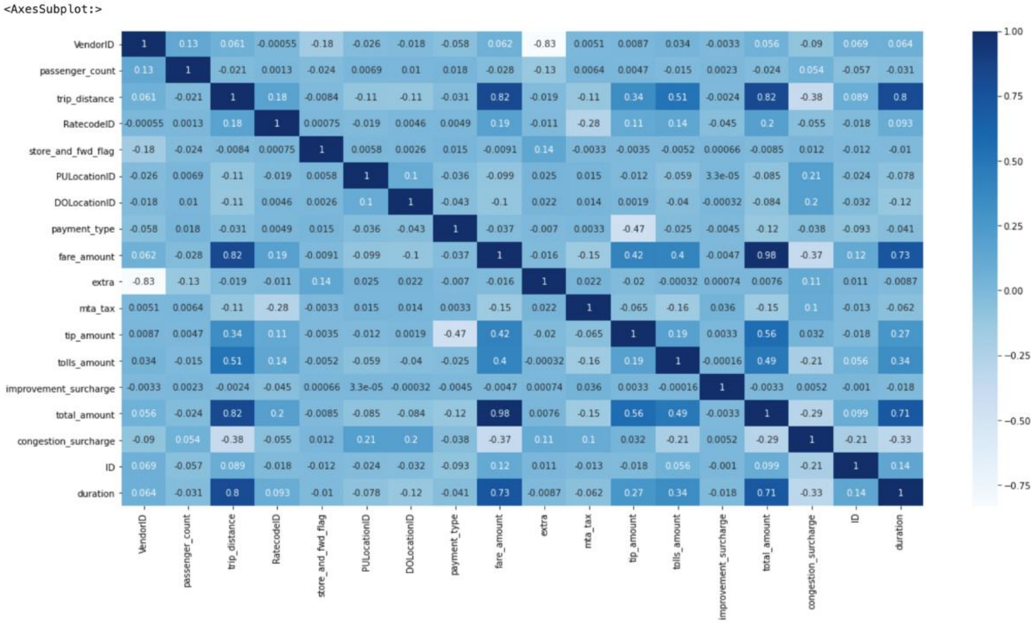
8. ANALYTICS, MACHINE LEARNING:

HEAT MAP OF CORRELATION MATRIX:

1. Heat map of the correlation matrix

[25]:
```python
#getting correlation matrix
corr = data.corr()
#plotting heat map
ax, fig = plt.subplots(figsize=[20,10])
sns.heatmap(corr, cmap="Blues", annot=True)
```

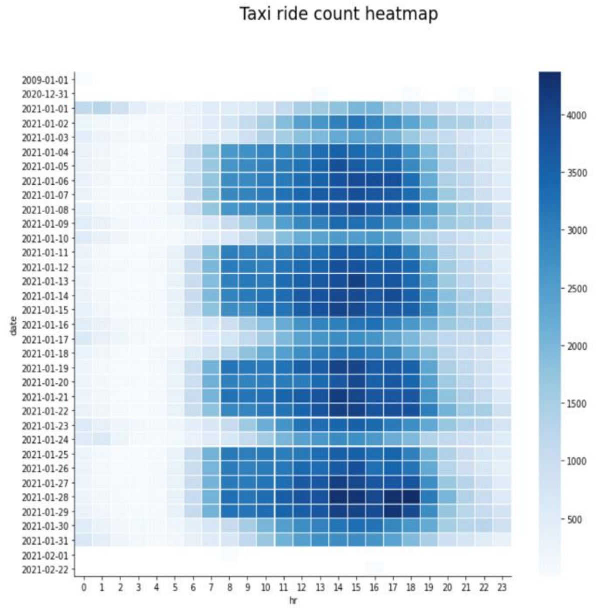
[25]: <AxesSubplot:>



We can deduce few finding from the correlation matrix heatmap:

1. Fare amout is correlated with trip duration and distance.
2. Tip amount is also correlated with fare amount. It can be obvious

TAXI RIDE HEAT MAP:

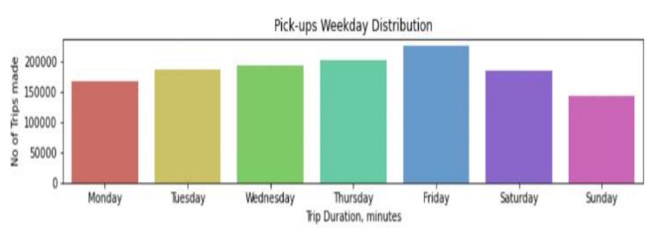


Taxi ride count heatmap shows some interesting obervations. By observing heatmap, we can see Taxi demard surge generally start increasing at 6 AM in the morning. Demand is at peak between 1 P.M. to 6 P.M. This pattern is followed on weekdays. On weekends, taxi demand seems to be less as compared to weekdays. Also, taxi demand can be seen more on Saturdays than Sundays. We can see on interesting observation from the ride count heatmap. Taxi demand surge can be seen from 12 A.M. to 3 A.M. on 1 Jan 2021. It must be becuase people are returning home after new year celebration/ party.

PICK-UPS HOURLY DISTRIBUTION:

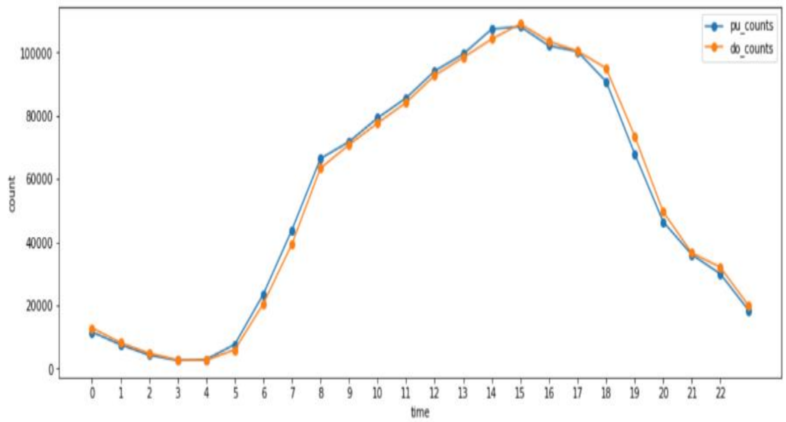
Pick-ups hourly distribution chart shows that taxi demand is less in the early morning. Demand start surging as the day passes. Demand is at peak in the afternoon. Then, it gradually slows down till the end of the day.

PICK-UPS MONTHLY DISTRIBUTION:



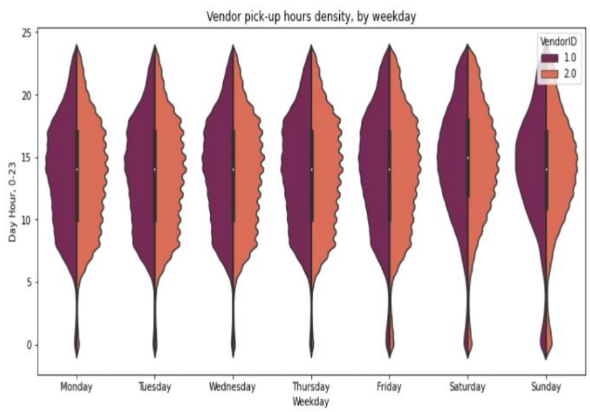
Pick-ups weekly distribution chart shows that taxi demand is the highest on Fridays. We can see the gradual increase in taxi demand from Monday to Friday. Tax demand is the lowest on Sundays.

9. EVALUATION & OPTIMIZATION:



- Taxi demand surge can be seen in the afternoon and is highest between 2 P.M. to 3 P.M.
- Taxi demand is lowest in the early morning between 3 A.M. to 4 A.M.

VENDOR PICK-UP HOURS DENSITY, BY WEEKDAY:



- As like our previous obervations, Violin plot also show that taxi demand surge is less in the early morning. Demand start surging as the day passes. Demand is at peak in the afternoon. Then, it gradually slows down till the end of the day.
- One interesting observation is that we can see surge in taxi demand in the midnight during the weekends. It could be possible because people go out on weekend for movies/ dinner.

10. RESULTS:

From the data that we have analyzed the following are the observations we are able to predict:

- ✓ It’s clear that tip amount is correlated with fare amount.
- ✓ It is also observed that trip amount is linked with the distance to be traveled and duration of the trip.
- ✓ With the help of a heat map it’s predictable that demand for taxis set to increase from 06:00 AM in the morning and its at peak during 01:00 PM to 06:00 PM which is during weekdays.
- ✓ During weekends demand for taxis seems to be more compared to weekdays.
- ✓ From the bar graph that’s plotted for Pick-ups hourly distribution depicted that demand for

taxi is set to be in increasing trend from 05:00 AM to 3:00 PM and then decreased gradually.

- ✓ Pick-ups hourly distribution is the highest 02:00 PM to 03:00 PM but almost the same in between 04:00 PM to 05:00 PM.
- ✓ Pick-ups hourly distribution is lowest during early morning i.e., in between 02:00 AM to 04:00 AM.
- ✓ Pick-ups Week-day distribution is high on Friday which is approximately 2.5 Lakh trips made. It can also be seen that demand is lowest on Sunday and it gradually increases from Monday.
- ✓ In the case of Vendor pickup hours density i.e., Violin plot it can be understood that the surge for taxi demand is less during early hours and more during midnights mainly on weekends.

11. FUTURE WORK:

- ✓ **WHAT WAS UNIQUE ABOUT THE DATA? DID YOU HAVE TO DEAL WITH IMBALANCE? WHAT DATA CLEANING DID YOU DO? OUTLIER TREATMENT? IMPUTATION?**

A few years ago, the New York City Taxi & Limousine Commission published a remarkable set of statistics, documenting 1.1 billion distinct taxi journeys made between 2009 and 2015. The information contains the GPS coordinates of the start and finish of each journey, providing a clear picture of where people went.

The provided data set has a sizable amount of imbalance. To modify the data, we used a variety of strategies. As they are not helpful, we have first eliminated the data for features that had null values. To make the data types for some features (tpep pickup datetime, tpep dropoff datetime) meaningful for analysis, we had to make the necessary corrections. We looked for and eliminated any abnormal data. For instance, there are some records in our data where the drop-off time is sooner than the pickup time. Ideally, this won't occur. These occurrences have been marked as anomalous, and they have been eliminated from the dataset. To make the features containing categorical variables understandable to machine learning algorithms, we converted them to numeric/Boolean format. Outlier treatment is explained by the aforementioned acts. By taking into account the mode value of each feature (VendorID, payment type, store and fwd flag, passenger count, RatecodeID), we have additionally imputed null values.

- ✓ **DID YOU CREATE ANY NEW ADDITIONAL FEATURES/VARIABLES?**

In order to aid in our study, we have built a few calculated fields from the provided data. A few of the variables include:

length: To record the duration of the complete trip, including the times of pickup and drop-off.

To save the time the pickup was completed, use the function hh pickup.

storing the hour at which the drop-off occurred using the variable hh dropoff.

dow pickup: Used to record the day of the week when the pickup took place.

dow dropoff: Used to save the day of the week that the excursion came to an end.

- ✓ **WHAT WAS THE PROCESS YOU USED FOR EVALUATION? WHAT WAS THE BEST RESULT?**

We carefully examined the data set we have selected and determined the busiest and least busy times of the day. Although we haven't specifically employed any models, we could use the study to implement the projections for fare surcharges in the future.

- ✓ **IS THERE BIAS IN YOUR WORK? WHAT WERE THE PROBLEMS YOU FACED? HOW DID YOU SOLVE THEM?**

The null values were eliminated, and for some of them, the median value was substituted. Although we first worried that the analysis would be impacted, eliminating null values actually made it simpler for us to move forward with the analytical portion. There are 6557 rows of anomalies, and they are all considered anomalous if the pickup time is later than the drop-off time. All of the anomalous data has been erased.

- ✓ **WHAT FUTURE WORK WOULD YOU LIKE TO DO?**

The work we've presented is primarily concerned with the potential surge charge application time of day. We anticipate being able to forecast when the surge charge will be at its highest and lowest points in the future. In order to make things clearer, it is also possible to anticipate the daily average surge charge. The best possible complement to the job we have already finished would be this.

- ✓ **INSTRUCTIONS FOR INDIVIDUALS THAT MAY WANT TO USE YOUR WORK**

REQUIREMENTS:

  - Amazon S3
  - Amazon Sagemaker
  - Python

- Jupyter Notebook

PACKAGES:

- Numpy
- Seaborn
- Pandas
- Matplotlib

INSTRUCTIONS FOR EXECUTION:

- The First step is to create an S3 bucket and upload the data that’s chosen. Here it’s the taxi_data.csv
- By creating a Jupyter Notebook instance in AWS Sagemaker data must be copied from the S3 bucket
- Then Upload “group-13-taxi-data-analysis.ipynb” in Jupyter Notebook
- Excute the cells one by one for desired results.