NYC Taxi Demand Prediction

BUSINESS UNDERSTANDING

EVALUATION

FUTURE IMPROVEMENTS

OBJECTIVE

DATA COLLECTION

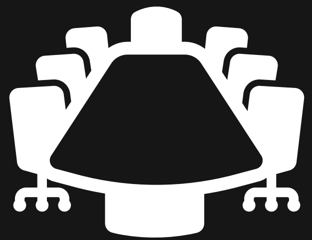
DATA PREPARATION

MODELS

OTHER FEATURES

INTRODUCTION

FEATURE UNDERSTANDING



INTRODUCTION

INTRODUCTION

Iconic history and significance
The concept of taxi medallions
Yellow cabs are concentrated in Manhattan, they serve all five boroughs of NYC.
Market Disruption, Competitive Challenges, Regulatory and Economic Impacts
Operational Efficiency, Economic Viability, Adaptation and Survival

OBJECTIVE

OBJECTIVE

- To analyze historical taxi trip data to understand the demand patterns across different times and locations in New York City.
- To develop a predictive model that can forecast taxi demand, helping in efficient fleet management and reducing passenger wait times.
- To utilize the predictive model to optimize taxi dispatches, thereby increasing operational efficiency and revenue.
- To contribute insights that can help improve urban transportation planning and reduce traffic congestion.
- To integrate both spatial and temporal data to provide precise demand forecasts that account for location-based and time-specific factors.
- To ensure taxis are used efficiently, contributing to a reduction in unnecessary idling and fuel consumption, thereby minimizing the environmental footprint.

BUSINESS UNDERSTANDING

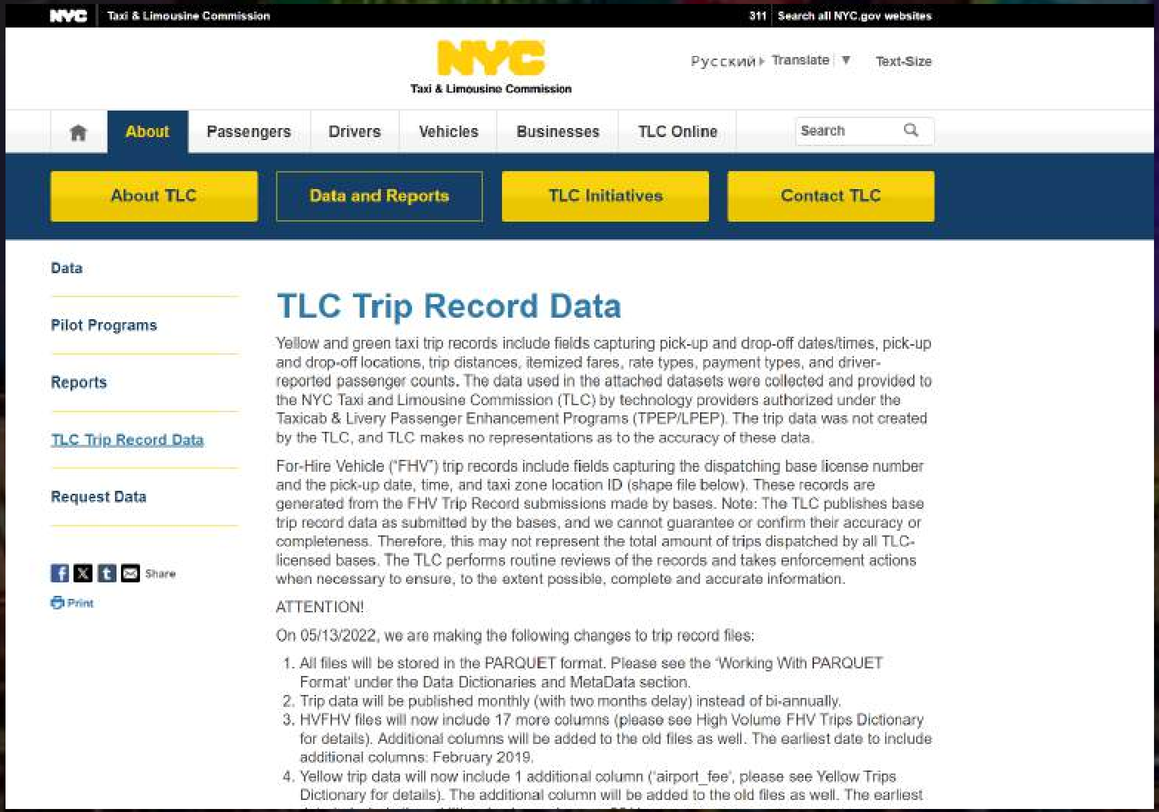
BUSINESS UNDERSTANDING

Industry Shift: The launch of ride-sharing services like Uber in 2011 introduced a new, flexible model of transportation, disrupting the traditional taxi service industry.
Decline in Taxi Use: Challenges faced by the traditional medallion (yellow) cabs with a significant decline in patronage due to the convenience and ease of booking offered by ride-sharing competitors.Survival **Through Adaptation**: Emphasis on the need for the traditional taxi industry to adapt by leveraging technology to optimize their operations and compete effectively in the changed landscape.Data-Driven Di**spatching:** Introducing the concept of predictive analytics as a solution for smarter dispatching, which could align taxi availability with fluctuating demand patterns.
Strategic Positioning of Cabs: Predictive insights could enable taxi dispatchers to strategically position cabs in high-demand areas, enhancing service responsiveness and customer satisfaction.
Profit Margin Revival: predictive modeling's potential to assist in critical decision-making that could lead to increased operational efficiency, better customer service, and ultimately, the revival of profit margins for the traditional taxi services.

DATA COLLECTION

DATA COLLECTION

The data used for building our solution was collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP).



NYC Taxi & Limousine Commission | 311 | Search all NYC.gov websites

NYC Taxi & Limousine Commission

Русский Translate Text-Size

About | Passengers | Drivers | Vehicles | Businesses | TLC Online | Search

About TLC | Data and Reports | TLC Initiatives | Contact TLC

- Data
- Pilot Programs
- Reports
- TLC Trip Record Data
- Request Data

Share

Print

TLC Trip Record Data

Yellow and green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the attached datasets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP). The trip data was not created by the TLC, and TLC makes no representations as to the accuracy of these data.

For-Hire Vehicle ("FHV") trip records include fields capturing the dispatching base license number and the pick-up date, time, and taxi zone location ID (shape file below). These records are generated from the FHV Trip Record submissions made by bases. Note: The TLC publishes base trip record data as submitted by the bases, and we cannot guarantee or confirm their accuracy or completeness. Therefore, this may not represent the total amount of trips dispatched by all TLC-licensed bases. The TLC performs routine reviews of the records and takes enforcement actions when necessary to ensure, to the extent possible, complete and accurate information.

ATTENTION!

On 05/13/2022, we are making the following changes to trip record files:

1. All files will be stored in the PARQUET format. Please see the 'Working With PARQUET Format' under the Data Dictionaries and MetaData section.
2. Trip data will be published monthly (with two months delay) instead of bi-annually.
3. HVFHV files will now include 17 more columns (please see High Volume FHV Trips Dictionary for details). Additional columns will be added to the old files as well. The earliest date to include additional columns: February 2019.
4. Yellow trip data will now include 1 additional column ('airport_fee', please see Yellow Trips Dictionary for details). The additional column will be added to the old files as well. The earliest

2015

January
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

February
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

March
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

April
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

May
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

June
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

July
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

August
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

September
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

October
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

November
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

December
- Yellow Taxi Trip Records (PARQUET)
- Green Taxi Trip Records (PARQUET)
- For-Hire Vehicle Trip Records (PARQUET)

FEATURE UNDERSTANDING

FEATURE UNDERSTANDING

temporal span: training - Jan 2015 ; testing - Jan 2016 geographical scope - all five boroughs of NYC

```
In [13]:  # Use 2015 data for training and 2016 data for testing
          df_train = df[df["year"] == 2015]
          df_test = df[df["year"] == 2016]

          # Create train and test datasets
          # Remove year column as it is no longer needed
          X_train = df_train.drop(columns=["pickup_count", "year"])
          y_train = df_train["pickup_count"]
          X_test = df_test.drop(columns=["pickup_count", "year"])
          y_test = df_test["pickup_count"]

          print("Train dataset:")
          display(X_train)
          display(y_train)

          print("Test dataset:")
          display(X_test)
          display(y_test)
```

Train dataset:

ear_sin	day_of_year_cos	time_10min_sin	time_10min_cos	day_of_week_1	day_of_week_2	day_of_week_3	day_of_week_4	day_of_week_5	day_of_week_6
336e-02	0.999852	0.000000	1.000000	False	False	True	False	False	False
336e-02	0.999852	0.043619	0.999048	False	False	True	False	False	False
336e-02	0.999852	0.087156	0.996195	False	False	True	False	False	False
336e-02	0.999852	0.130526	0.991445	False	False	True	False	False	False
336e-02	0.999852	0.173648	0.984808	False	False	True	False	False	False
...	...	...	...	...	...	...	...	...	...
491e-16	1.000000	-0.216440	0.976296	False	False	True	False	False	False
491e-16	1.000000	-0.173648	0.984808	False	False	True	False	False	False
491e-16	1.000000	-0.130526	0.991445	False	False	True	False	False	False
491e-16	1.000000	-0.087156	0.996195	False	False	True	False	False	False
491e-16	1.000000	-0.043619	0.999048	False	False	True	False	False	False

```
In [8]: # Join the weather data with the main dataframe

df = pd.merge(df, df_weather, how="left", on="date")

print("Final dataframe with weather data:")
display(df)
display(df.info())
```

Final dataframe with weather data:

	LocationID	year	day_of_year	time_10min	pickup_count	date	is_holiday	day_of_week	temperature_max	temperature_avg	...	humidity_max
0	1	2015	1	0	0	2015-01-01	True	3	38	33.3	...	46
1	1	2015	1	1	0	2015-01-01	True	3	38	33.3	...	46
2	1	2015	1	2	0	2015-01-01	True	3	38	33.3	...	46
3	1	2015	1	3	0	2015-01-01	True	3	38	33.3	...	46
4	1	2015	1	4	0	2015-01-01	True	3	38	33.3	...	46
...	...	...	...	...	...	...	...	...	...	...	...	...
27856795	265	2016	365	139	4	2016-12-30	False	4	42	38.5	...	82
27856796	265	2016	365	140	1	2016-12-30	False	4	42	38.5	...	82
27856797	265	2016	365	141	4	2016-12-30	False	4	42	38.5	...	82
27856798	265	2016	365	142	1	2016-12-30	False	4	42	38.5	...	82
27856799	265	2016	365	143	5	2016-12-30	False	4	42	38.5	...	82

27856800 rows × 24 columns

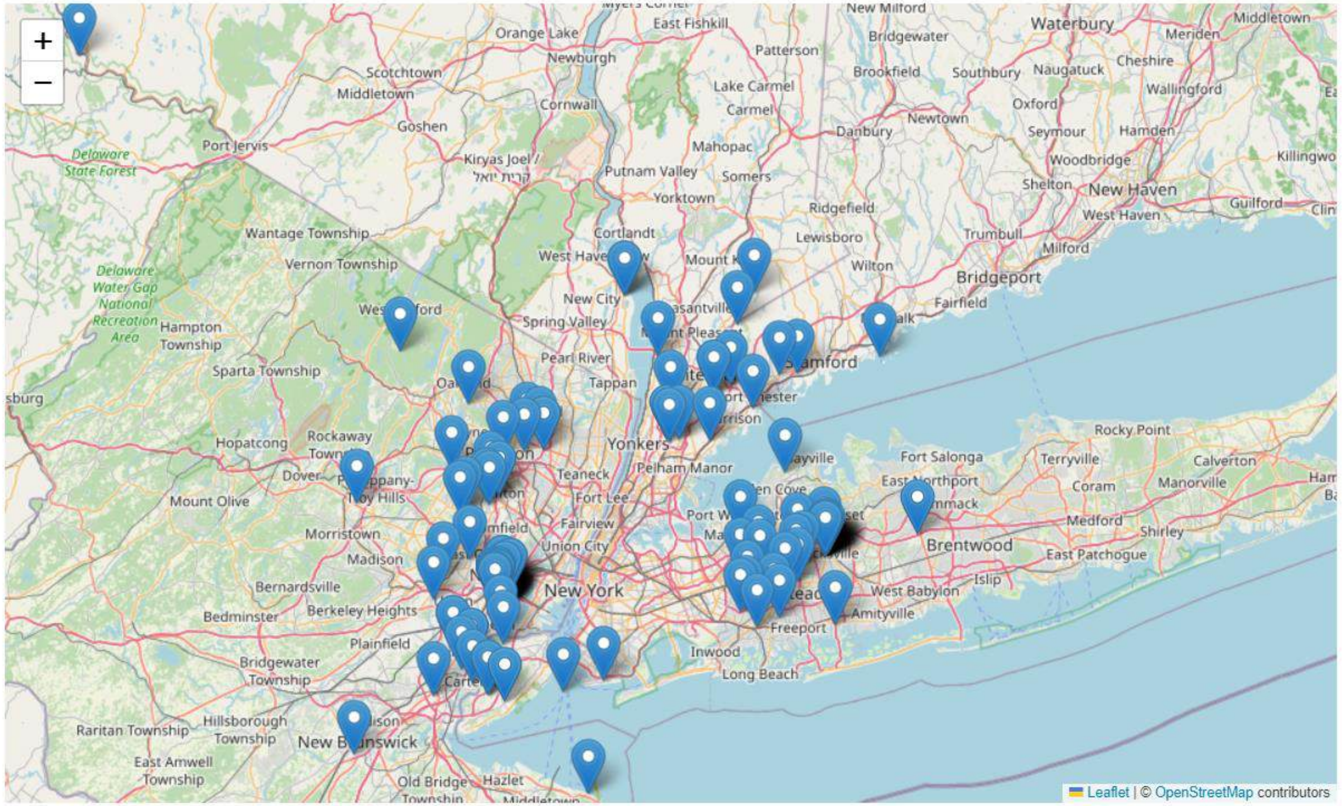
VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger	trip_distan	pickup_lon	pickup_lati	RateCodel	store_and_	dropoff_lor	dropoff_lat	payment_t	fare_amou	extra	mta_tax	tip_amoun	tolls_amou	improveme	total_amo
2	1/15/2015 19:05	1/15/2015 19:23	1	1.59	-73.9939	40.750110	1	N	-73.9748	40.750617	1	12	1	0.5	3.25	0	0.3	17.05
1	1/10/2015 20:33	1/10/2015 20:53	1	3.3	-74.0016	40.72424	1	N	-73.9944	40.759109	1	14.5	0.5	0.5	2	0	0.3	17.8
1	1/10/2015 20:33	1/10/2015 20:43	1	1.8	-73.9633	40.802787	1	N	-73.9518	40.824413	2	9.5	0.5	0.5	0	0	0.3	10.8
1	1/10/2015 20:33	1/10/2015 20:35	1	0.5	-74.0091	40.713817	1	N	-74.0043	40.719985	2	3.5	0.5	0.5	0	0	0.3	4.8
1	1/10/2015 20:33	1/10/2015 20:52	1	3	-73.9712	40.762428	1	N	-74.0042	40.742652	2	15	0.5	0.5	0	0	0.3	16.3
1	1/10/2015 20:33	1/10/2015 20:53	1	9	-73.8744	40.77405	1	N	-73.987	40.758193	1	27	0.5	0.5	6.7	5.33	0.3	40.33
1	1/10/2015 20:33	1/10/2015 20:58	1	2.2	-73.9833	40.726009	1	N	-73.9925	40.74963	2	14	0.5	0.5	0	0	0.3	15.3
1	1/10/2015 20:33	1/10/2015 20:42	3	0.8	-74.0027	40.734142	1	N	-73.995	40.726325	1	7	0.5	0.5	1.66	0	0.3	9.96
1	1/10/2015 20:33	1/10/2015 21:11	3	18.2	-73.783	40.644355	2	N	-73.9876	40.759357	2	52	0	0.5	0	5.33	0.3	58.13
1	1/10/2015 20:33	1/10/2015 20:40	2	0.9	-73.9856	40.767948	1	N	-73.9859	40.759365	1	6.5	0.5	0.5	1.55	0	0.3	9.35
1	1/10/2015 20:33	1/10/2015 20:41	1	0.9	-73.9886	40.723102	1	N	-74.0044	40.728584	1	7	0.5	0.5	1.66	0	0.3	9.96
1	1/10/2015 20:33	1/10/2015 20:43	1	1.1	-73.9938	40.751419	1	N	-73.9674	40.757217	1	7.5	0.5	0.5	1	0	0.3	9.8
1	1/10/2015 20:33	1/10/2015 20:35	1	0.3	-74.0084	40.704376	1	N	-74.0098	40.707725	2	3	0.5	0.5	0	0	0.3	4.3
1	1/10/2015 20:33	1/10/2015 21:03	1	3.1	-73.9739	40.760448	1	N	-73.9973	40.735210	1	19	0.5	0.5	3	0	0.3	23.3
1	1/10/2015 20:33	1/10/2015 20:39	1	1.1	-74.0067	40.731777	1	N	-73.9952	40.739894	2	6	0.5	0.5	0	0	0.3	7.3
2	1/15/2015 19:05	1/15/2015 19:32	1	2.38	-73.9764	40.739810	1	N	-73.984	40.757888	1	16.5	1	0.5	4.38	0	0.3	22.68
2	1/15/2015 19:05	1/15/2015 19:21	5	2.83	-73.9687	40.754245	1	N	-73.9551	40.786857	2	12.5	1	0.5	0	0	0.3	14.3
2	1/15/2015 19:05	1/15/2015 19:28	5	8.33	-73.8631	40.769580	1	N	-73.9527	40.785781	1	26	1	0.5	8.08	5.33	0.3	41.21
2	1/15/2015 19:05	1/15/2015 19:20	1	2.37	-73.9455	40.779422	1	N	-73.9809	40.786083	1	11.5	1	0.5	0	0	0.3	13.3
2	1/15/2015 19:05	1/15/2015 19:20	2	7.13	-73.8745	40.774009	1	N	-73.9524	40.718589	1	21.5	1	0.5	4.5	0	0.3	27.8
2	1/15/2015 19:05	1/15/2015 19:31	1	3.6	-73.9766	40.751895	1	N	-73.9989	40.714595	2	17.5	1	0.5	0	0	0.3	19.3
2	1/15/2015 19:05	1/15/2015 19:10	1	0.89	-73.995	40.745079	1	N	-73.9999	40.734649	1	5.5	1	0.5	1.62	0	0.3	8.92
2	1/15/2015 19:05	1/15/2015 19:10	1	0.96	-74.0009	40.747062	1	N	-74.0036	40.735511	1	5.5	1	0.5	1.3	0	0.3	8.6
2	1/15/2015 19:05	1/15/2015 19:12	2	1.25	-74.0028	40.717891	1	N	-74.0079	40.704219	1	6.5	1	0.5	1.5	0	0.3	9.8
2	1/15/2015 19:05	1/15/2015 19:22	5	2.11	-73.9975	40.736362	1	N	-73.9782	40.761856	1	11.5	1	0.5	2.5	0	0.3	15.8

DATA CLEANING

1.Pickup locations:

```
for i,j in sample_locations.iterrows():
    if int(j['pickup_latitude'])!=0:
        folium.Marker(list((j['pickup_latitude'],j['pickup_longitude']))).add_to(map_osm)
map_osm
```

Out[8]:



2. Drop-off locations:

2. Drop-off latitude and longitude.

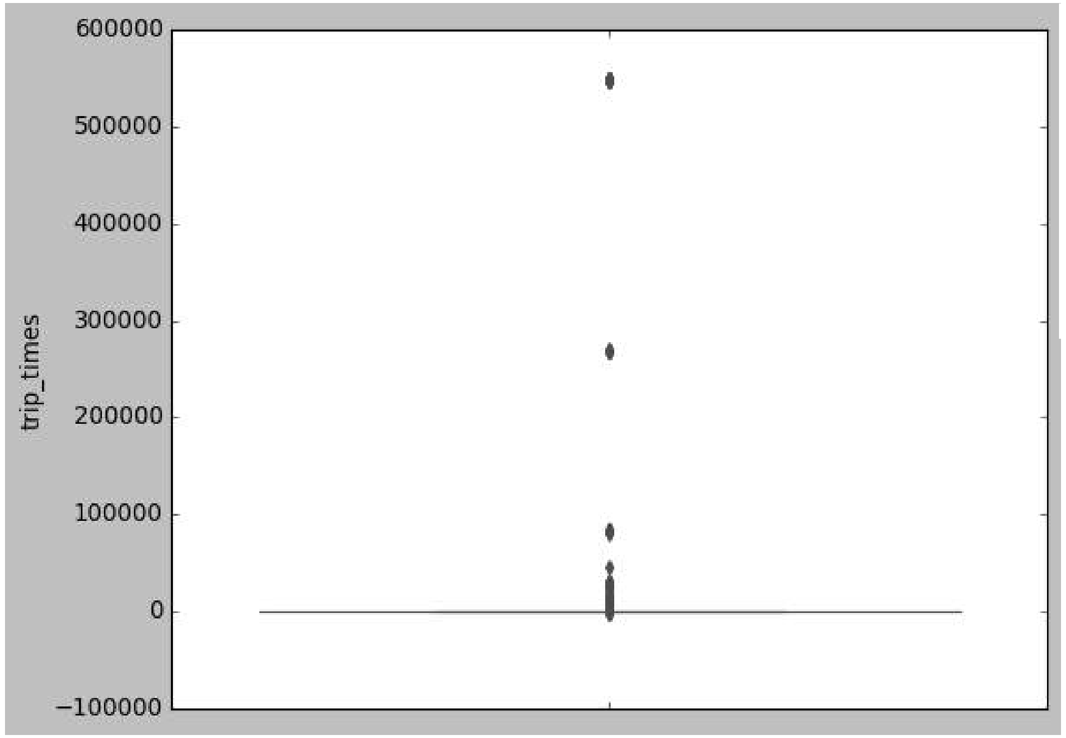
In [9]:

```python
#ploting pickup coordinates which are outside the boundary box of newyork
#we will collect all the points outside the bounding the box of newyork city to outlier_location
outlier_locations=month[((month.dropoff_longitude<=-74.15)| (month.dropoff_latitude<=40.5774)| \
                   (month.dropoff_longitude>=-73.7004)|(month.dropoff_latitude>=40.9176))]

map_osm = folium.Map(location=[40.734685, -73.990372], tiles='OpenStreetMap')

sample_locations=outlier_locations.head(10000)
for i,j in sample_locations.iterrows():
    if int(j['dropoff_latitude'])!=0:
        folium.Marker(list((j['dropoff_latitude'],j['dropoff_longitude']))).add_to(map_osm)
map_osm
```

Out[9]:

3.Trip Durations:

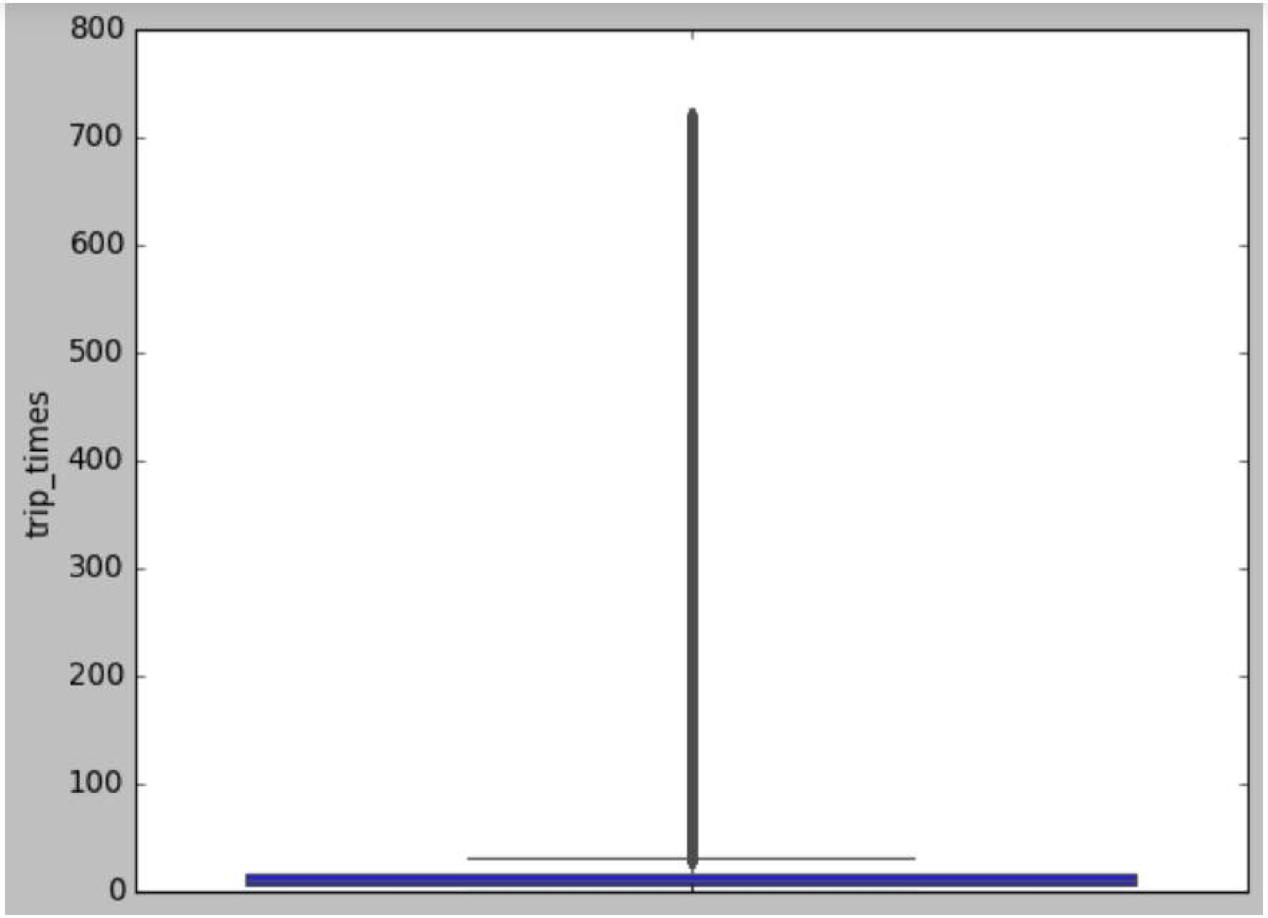
According to NYC Taxi & Limousine Commision Regulations the maximum allowed trip duration in a 24 hour interval is 12 hours. The timestamps are converted to unix so as to get duration (trip-time) & speed also pickup-times in unix are used while binning.




```
0 percentile value is -1211.0166666666667
10 percentile value is 3.8333333333333335
20 percentile value is 5.383333333333334
30 percentile value is 6.816666666666666
40 percentile value is 8.3
50 percentile value is 9.95
60 percentile value is 11.866666666666667
70 percentile value is 14.283333333333333
80 percentile value is 17.633333333333333
90 percentile value is 23.45
100 percentile value is  548555.6333333333
```

```
90 percentile value is 23.45
91 percentile value is 24.35
92 percentile value is 25.383333333333333
93 percentile value is 26.55
94 percentile value is 27.933333333333334
95 percentile value is 29.583333333333332
96 percentile value is 31.683333333333334
97 percentile value is 34.46666666666667
98 percentile value is 38.71666666666667
99 percentile value is 46.75
100 percentile value is  548555.6333333333
```

Trip time after removing outliers



4. Speed

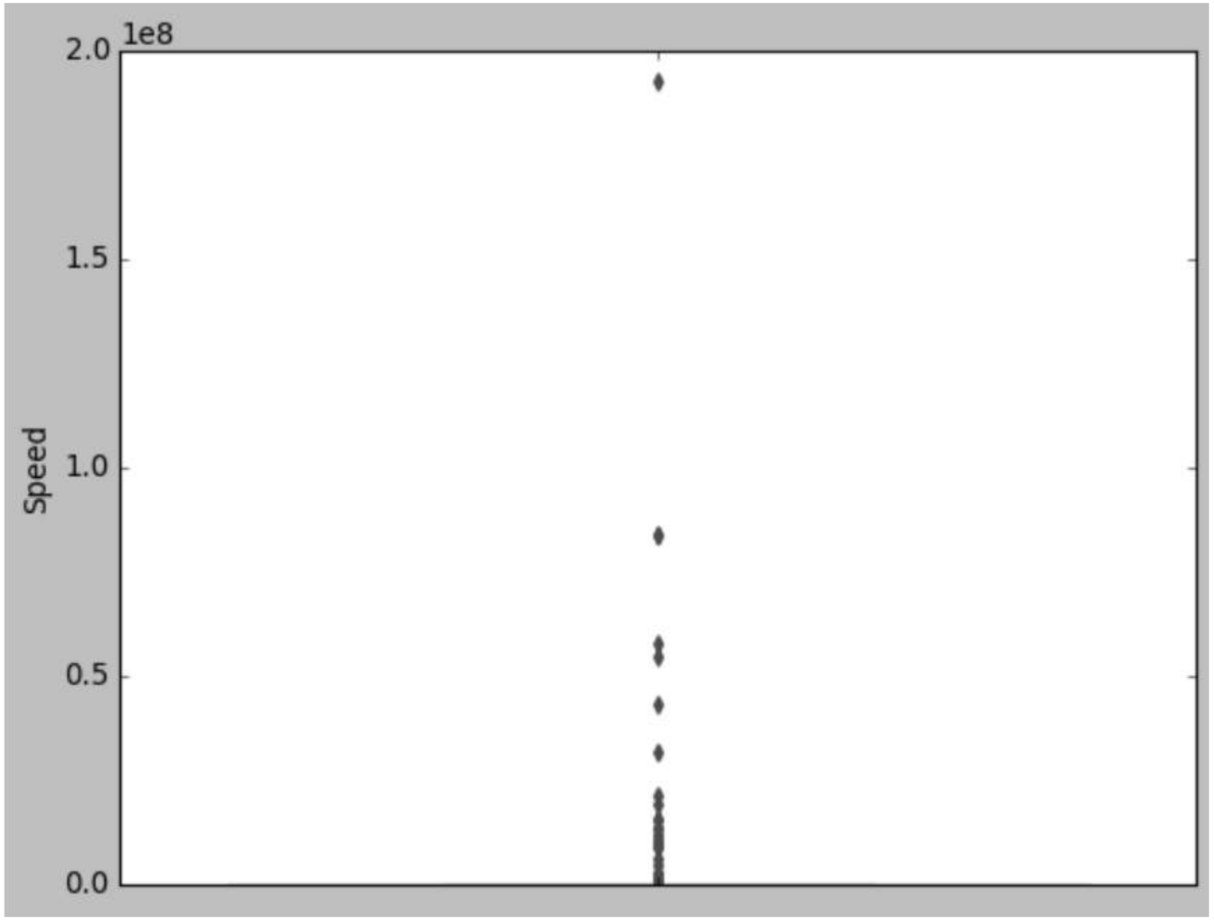
```
0 percentile value is 0.0
10 percentile value is 6.409495548961425
20 percentile value is 7.80952380952381
30 percentile value is 8.929133858267717
40 percentile value is 9.98019801980198
50 percentile value is 11.06865671641791
60 percentile value is 12.286689419795222
70 percentile value is 13.796407185628745
80 percentile value is 15.963224893917962
90 percentile value is 20.186915887850468
100 percentile value is  192857142.85714284
```

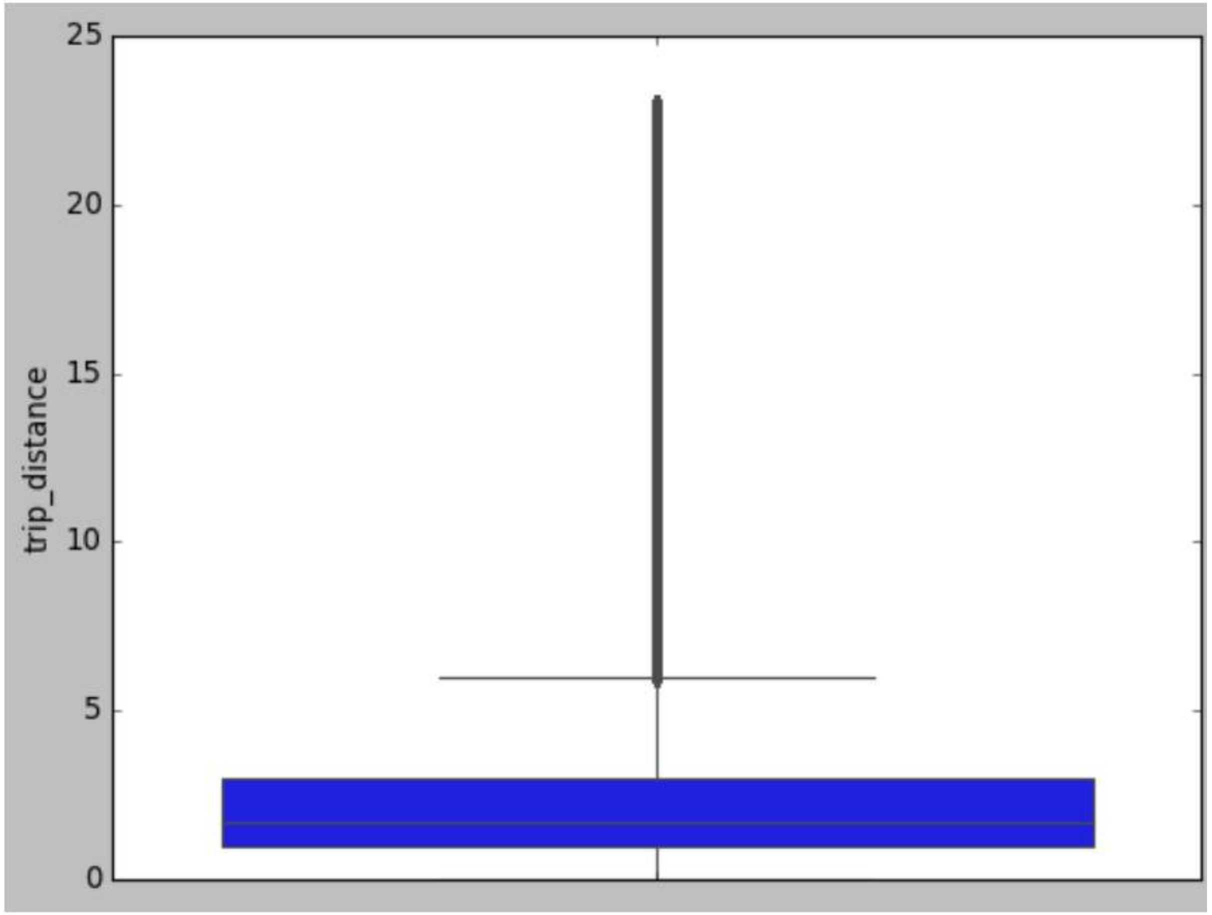
```
90 percentile value is 20.186915887850468
91 percentile value is 20.91645569620253
92 percentile value is 21.752988047808763
93 percentile value is 22.721893491124263
94 percentile value is 23.844155844155843
95 percentile value is 25.182552504038775
96 percentile value is 26.80851063829787
97 percentile value is 28.84304932735426
98 percentile value is 31.591128254580514
99 percentile value is 35.7513566847558
100 percentile value is  192857142.85714284
```

```
99.0 percentile value is 35.7513566847558
99.1 percentile value is 36.31084727468969
99.2 percentile value is 36.91470054446461
99.3 percentile value is 37.588235294117645
99.4 percentile value is 38.33035714285714
99.5 percentile value is 39.17580340264651
99.6 percentile value is 40.15384615384615
99.7 percentile value is 41.338301043219076
99.8 percentile value is 42.86631016042781
99.9 percentile value is 45.3107822410148
100 percentile value is  192857142.85714284
```

In [29]:
```
#avg.speed of cabs in New-York
sum(frame_with_durations_modified["Speed"]) / float(len(frame_with_durations_modified["Speed"]))
```

Out[29]:
```
12.450173996028015
```





Before removal of outliers

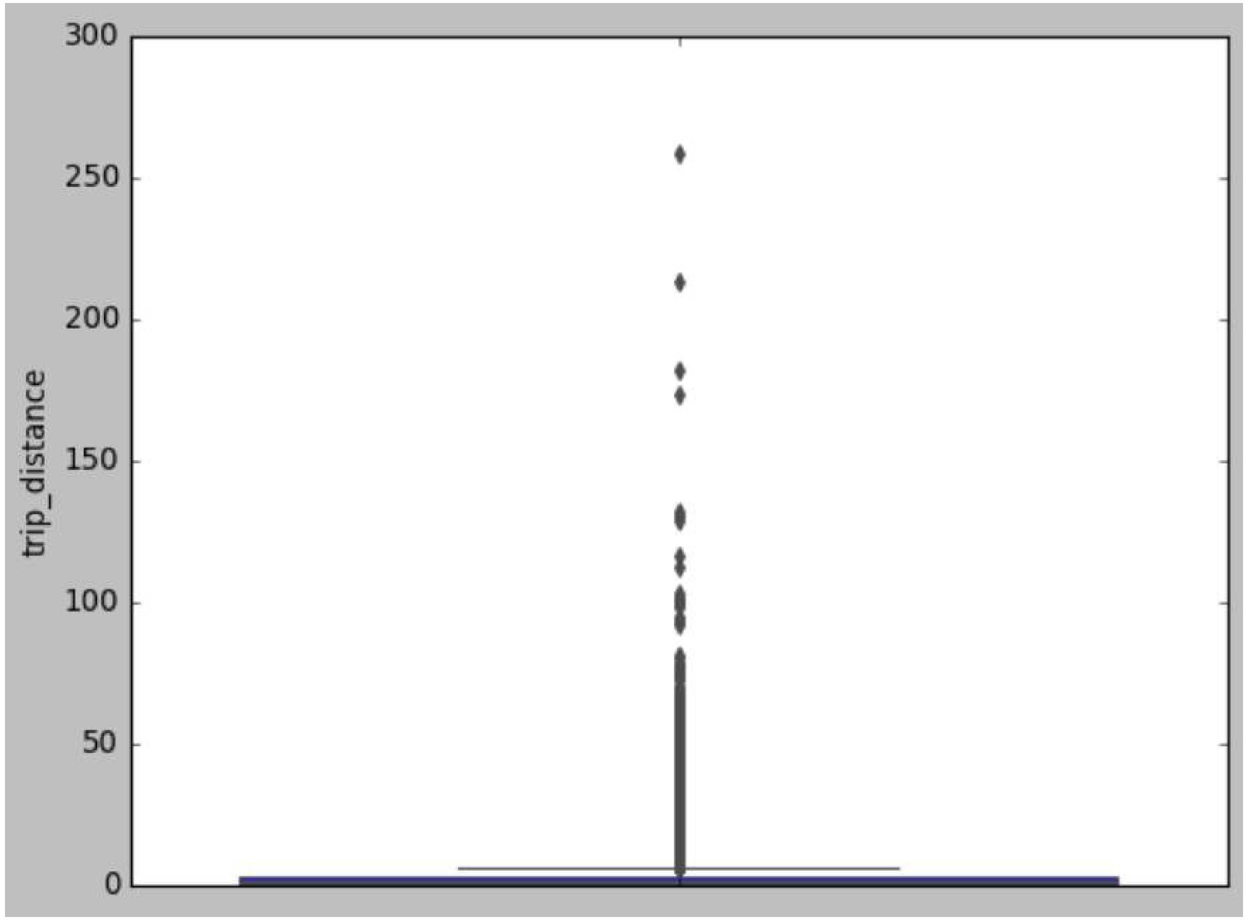
After removal of outliers

5. Trip Distance.

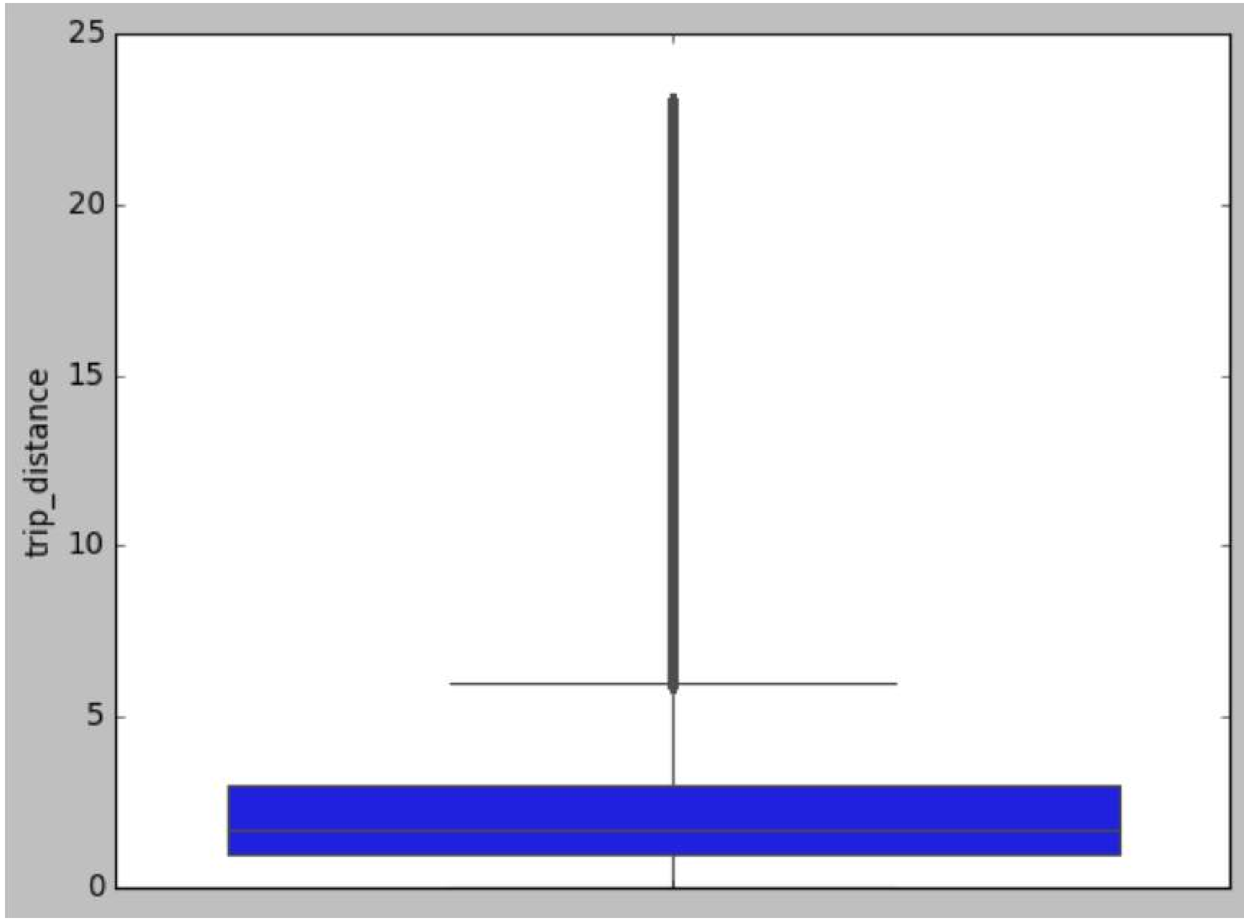
```
0 percentile value is 0.01
10 percentile value is 0.66
20 percentile value is 0.9
30 percentile value is 1.1
40 percentile value is 1.39
50 percentile value is 1.69
60 percentile value is 2.07
70 percentile value is 2.6
80 percentile value is 3.6
90 percentile value is 5.97
100 percentile value is  258.9
```

```
90 percentile value is 5.97
91 percentile value is 6.45
92 percentile value is 7.07
93 percentile value is 7.85
94 percentile value is 8.72
95 percentile value is 9.6
96 percentile value is 10.6
97 percentile value is 12.1
98 percentile value is 16.03
99 percentile value is 18.17
100 percentile value is  258.9
```

```
Speed at 99.0th percentile: 18.17
Speed at 99.1th percentile: 18.37
Speed at 99.19999999999999th percentile: 18.6
Speed at 99.29999999999998th percentile: 18.83
Speed at 99.39999999999998th percentile: 19.13
Speed at 99.49999999999997th percentile: 19.5
Speed at 99.59999999999997th percentile: 19.96
Speed at 99.69999999999996th percentile: 20.5
Speed at 99.79999999999995th percentile: 21.22
Speed at 99.89999999999995th percentile: 22.57
Speed at 99.99999999999994th percentile: 258.899999
```

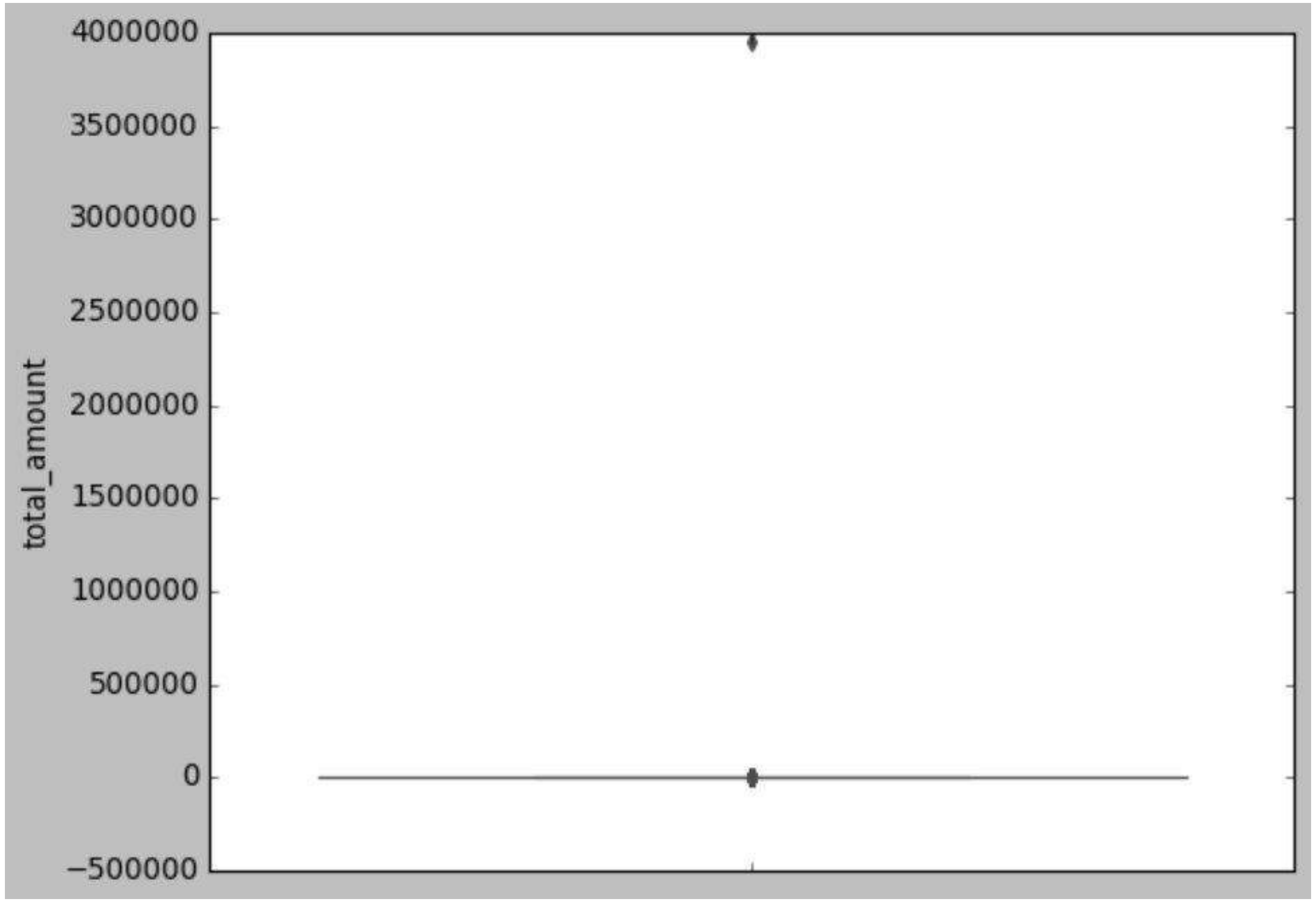


Before removal of outliers



After removal of outliers

6. Total Fare



4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
−500000
total_amount

```
0 percentile value is -242.55
10 percentile value is 6.3
20 percentile value is 7.8
30 percentile value is 8.8
40 percentile value is 9.8
50 percentile value is 11.16
60 percentile value is 12.8
70 percentile value is 14.8
80 percentile value is 18.3
90 percentile value is 25.8
100 percentile value is  3950611.6
```

```
Speed at 99.0th percentile: 66.13
Speed at 99.1th percentile: 68.13
Speed at 99.19999999999999th percentile: 69.6
Speed at 99.29999999999998th percentile: 69.6
Speed at 99.39999999999998th percentile: 69.73
Speed at 99.49999999999997th percentile: 69.75
Speed at 99.59999999999997th percentile: 69.76
Speed at 99.69999999999996th percentile: 72.58
Speed at 99.79999999999995th percentile: 75.35
Speed at 99.89999999999995th percentile: 88.27223999984562
Speed at 99.99999999999994th percentile: 3950611.5705955215
```

```
90 percentile value is 25.8
91 percentile value is 27.3
92 percentile value is 29.3
93 percentile value is 31.8
94 percentile value is 34.8
95 percentile value is 38.53
96 percentile value is 42.6
97 percentile value is 48.13
98 percentile value is 58.13
99 percentile value is 66.13
100 percentile value is  3950611.6
```

Remove all outliers/erronous points.

```
Removing outliers in the month of Jan-2015
---
number of pickup records= 12748986
number of outlier coordinates lying outside NY boundries: 293919
Number of outliers from the trip time analysis: 23889
Number of outliers from trip distance analysis: 92597
Number of outlier from speed analysis: 24473
Number of outlier from fare analysis: 5275
Total outliers removed 377910
---
fraction of data points that remains after removing outliers 0.9703576425607495
```

DATA PREPARATION

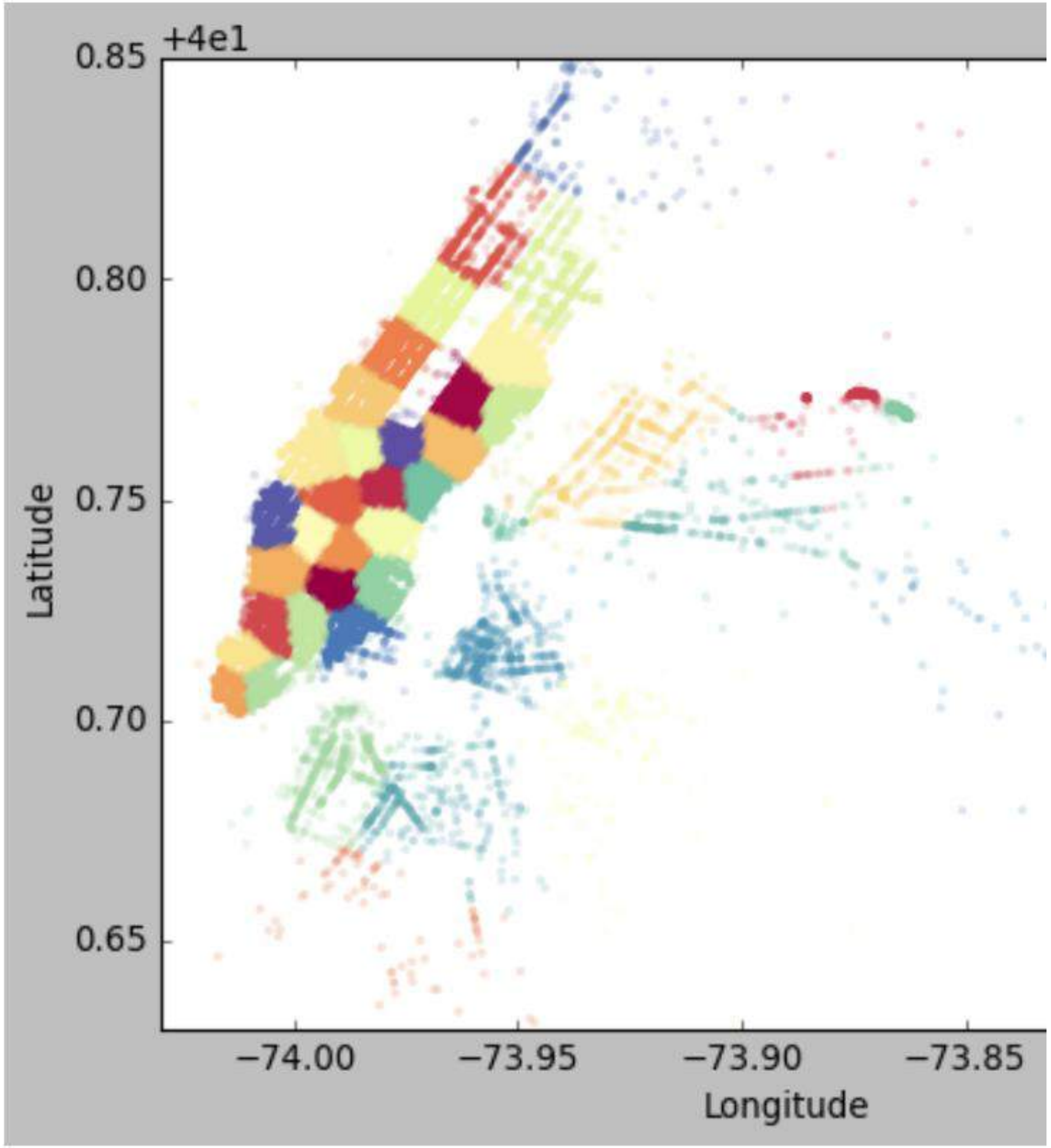
DATA PREPARATION

1. Clustering and segmentation :

   Creating cluster using k-means method.

   The regions with more number of pickups form smaller and dense clusters whereas regions with lesser number of pickups get connected into larger and loose clusters.

   We need to find the optimal number of clusters needed to divide in the geo area, for that we need to check minimum inner cluster should be less and out cluster should be more.

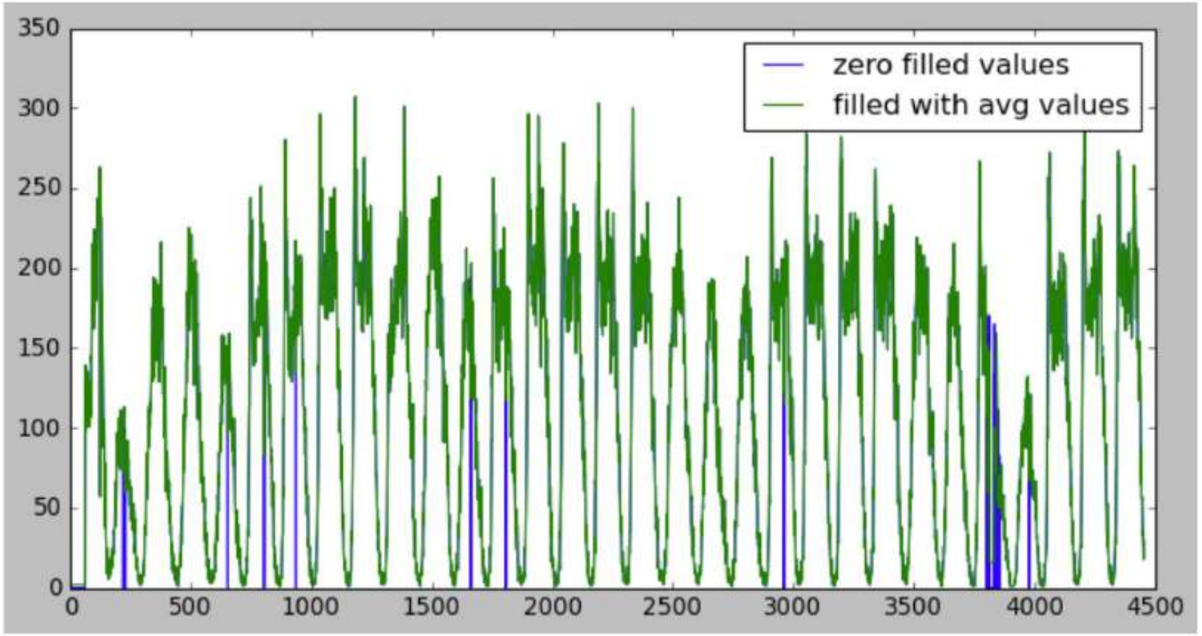



```
On choosing a cluster size of  10
Avg. Number of Clusters within the vicinity (i.e. intercluster-distance < 2): 2.0
Avg. Number of Clusters outside the vicinity (i.e. intercluster-distance > 2): 7.0
Min inter-cluster distance =  0.9942822667922672
---
On choosing a cluster size of  20
Avg. Number of Clusters within the vicinity (i.e. intercluster-distance < 2): 5.0
Avg. Number of Clusters outside the vicinity (i.e. intercluster-distance > 2): 14.0
Min inter-cluster distance =  0.6444725834028739
---
On choosing a cluster size of  30
Avg. Number of Clusters within the vicinity (i.e. intercluster-distance < 2): 9.0
Avg. Number of Clusters outside the vicinity (i.e. intercluster-distance > 2): 21.0
Min inter-cluster distance =  0.47920626820356643
---
On choosing a cluster size of  40
Avg. Number of Clusters within the vicinity (i.e. intercluster-distance < 2): 11.0
Avg. Number of Clusters outside the vicinity (i.e. intercluster-distance > 2): 29.0
Min inter-cluster distance =  0.36064577963428435
---
On choosing a cluster size of  50
Avg. Number of Clusters within the vicinity (i.e. intercluster-distance < 2): 13.0
Avg. Number of Clusters outside the vicinity (i.e. intercluster-distance > 2): 37.0
Min inter-cluster distance =  0.37726530352041876
```

```
Return with trip times..
Remove outliers..
number of pickup records= 10906858
number of outlier coordinates lying outside NY boundries:
Number of outliers from the trip time analysis: 27190
Number of outliers from trip distance analysis: 79742
Number of outlier from speed analysis: 21047
Number of outlier from fare analysis: 4991
Total outliers removed 297784
---
Estimating clusters..
Final groupbying..
```

pickup_cluster	pickup_bins	trip_distance
	63	93
	64	174
0	65	208
	66	174

2.SMOOTHING:



```
---------------------------------------------------------------------
for the  3 th cluster number of 10min intavels with zero pickups:  40
---------------------------------------------------------------------
for the  4 th cluster number of 10min intavels with zero pickups:  270
---------------------------------------------------------------------
for the  5 th cluster number of 10min intavels with zero pickups:  44
---------------------------------------------------------------------
for the  6 th cluster number of 10min intavels with zero pickups:  41
---------------------------------------------------------------------
for the  7 th cluster number of 10min intavels with zero pickups:  35
---------------------------------------------------------------------
for the  8 th cluster number of 10min intavels with zero pickups:  696
---------------------------------------------------------------------
for the  9 th cluster number of 10min intavels with zero pickups:  39
---------------------------------------------------------------------
for the  10 th cluster number of 10min intavels with zero pickups:  37
---------------------------------------------------------------------
for the  11 th cluster number of 10min intavels with zero pickups:  93
---------------------------------------------------------------------
for the  12 th cluster number of 10min intavels with zero pickups:  31
---------------------------------------------------------------------
for the  13 th cluster number of 10min intavels with zero pickups:  28
---------------------------------------------------------------------
for the  14 th cluster number of 10min intavels with zero pickups:  36
---------------------------------------------------------------------
for the  15 th cluster number of 10min intavels with zero pickups:  31
---------------------------------------------------------------------
for the  16 th cluster number of 10min intavels with zero pickups:  56
---------------------------------------------------------------------
for the  17 th cluster number of 10min intavels with zero pickups:  37
---------------------------------------------------------------------
for the  18 th cluster number of 10min intavels with zero pickups:  24
---------------------------------------------------------------------
for the  19 th cluster number of 10min intavels with zero pickups:  40
---------------------------------------------------------------------
```

3.Time Series and Fourier Transformers:

We can see that the number of pickups in a month in every cluster form a repeating pattern.

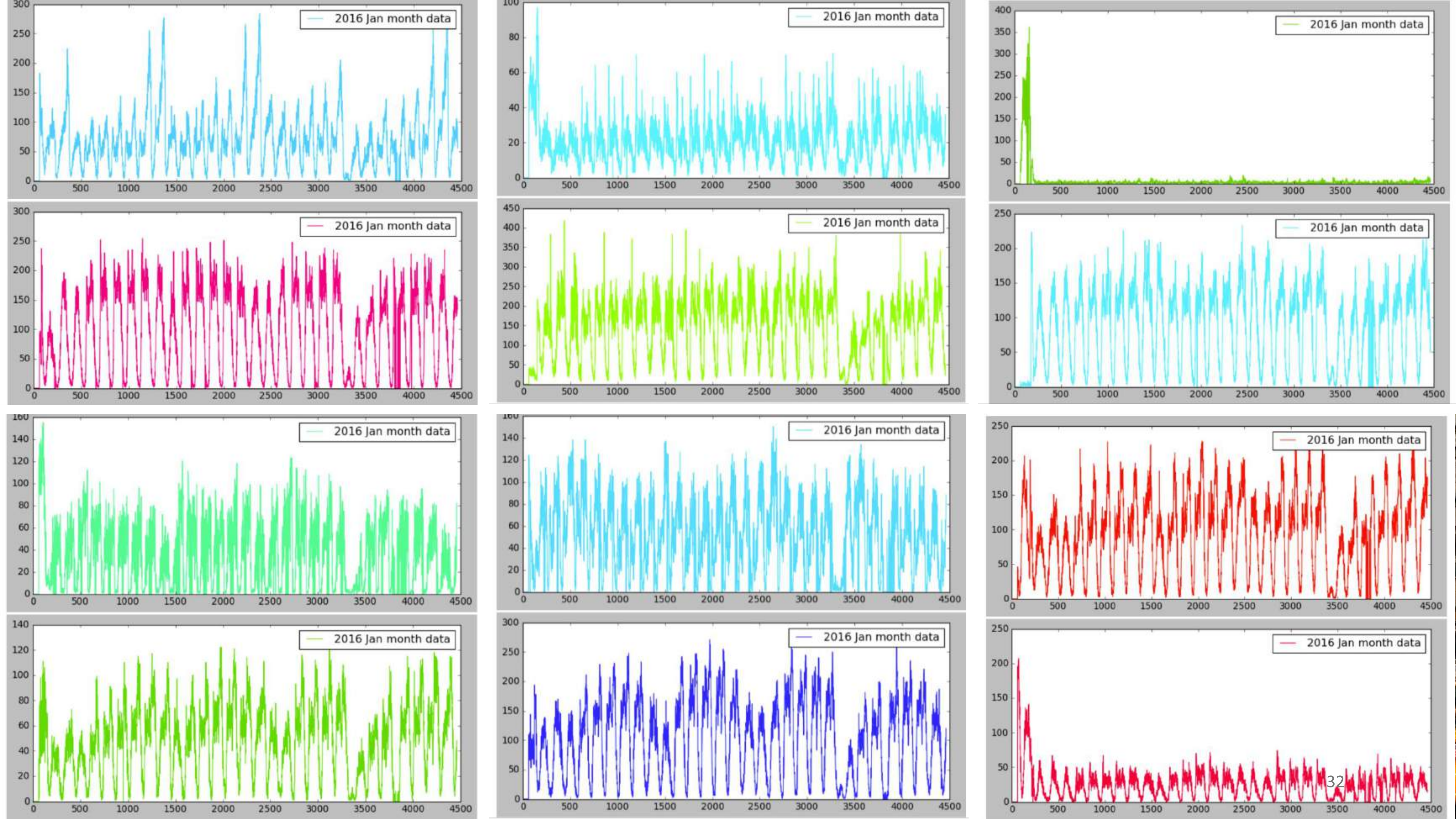
Fourier transform lets us represent our pattern from time domain (number of pickups per time) to frequency domain(can be viewed as number pickup bins with highest number of pickups).

For each cluster there exists a pattern and using the Fourier transform we can deduce the top frequencies and amplitudes of sine waves which compose our pattern from cluster and use them as features.

The frequencies and amplitudes of a cluster are indicative of demand in that cluster. So they can be fed into the model for prediction of number of pickups.

1. The pattern whose repetition is very high will have a high frequency component and vice versa.

2. There are high frequency in morning and evening time durations as the pick-ups are high during peak hours. The same applies to day and night but with less frequencies.


2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data
2016 Jan month data

MODELS

MODELS

BASELINE MODELS

REGRESSOR MODELS

LINEAR REGRESSOR

XGBoost Regressor

RANDOM FOREST

BASELINE MODELS

Baseline model:
1. Using Ratios of the 2016 data to the 2015 data i.e. Ratio= P_2016/P_2015
2. Using Previous known values of the 2016 data itself to predict the future values

Our 3 choices were **Simple Moving Averages**, **Weighted Moving Averages** and **Exponential Weighted Moving Average**. For each of them, Ratios and Predictions were calculated for P_2015 and P_2016.

Simple Moving Averages:
The First Model used is the Moving Averages Model which uses the previous n values in order to predict the next value

The First Model used is the Moving Averages Model which uses the previous n values in order to predict the next value

Using Ratio Values - $Rt=(Rt-1+Rt-2+Rt-3....Rt-n)/n$

Exponential_moving_average:

Through weighted averaged we have satisfied the analogy of giving higher weights to the latest value and decreasing weights to the subsequent ones but we still do not know which is the correct weighting scheme as there are infinetly many possibilities in which we can assign weights in a non-increasing order and tune the the hyperparameter window-size. To simplify this process we use Exponential Moving Averages which is a more logical way towards assigning weights and at the same time also using an optimal window-size.

In exponential moving averages we use a single hyperparameter alpha which is a value between 0 & 1 and based on the value of the hyperparameter alpha the weights and the window sizes are configured.

$$R'_t=\alpha*R_{t-1}+(1-\alpha)*R'_{t-1}$$

```
Error Metric Matrix (Forecasting Methods) - MAPE & MSE
--------------------------------------------------------------------------------------------------------
Moving Averages (Ratios) -                          MAPE:  0.3062982780952648       MSE:  3216.790176971326
Moving Averages (2016 Values) -                     MAPE:  0.15087674445032245       MSE:  212.38060035842295
--------------------------------------------------------------------------------------------------------
Weighted Moving Averages (Ratios) -                 MAPE:  0.31414114299290913      MSE:  2807.2774305555554
Weighted Moving Averages (2016 Values) -            MAPE:  0.14320561490386557      MSE:  192.9197748655914
--------------------------------------------------------------------------------------------------------
Exponential Moving Averages (Ratios) -           MAPE:  0.3209817925518331      MSE:  2809.560047043011
Exponential Moving Averages (2016 Values) -      MAPE:  0.1428974374720102      MSE:  191.02174059139784
```

Plese Note:- The above comparisons are made using Jan 2015 and Jan 2016 only

REGRESSOR MODELS

Regressor:
After done with baseline modeling, we have good features which helps us to build a good model prediction

Exponential weighted moving averages gives the best forecasting among the rest. We will use this as a feature while building the regression model along with others we got from data preparation stage.

	ft_5	ft_4	ft_3	ft_2	ft_1	lat	lon	weekday	exp_avg
0	23	26	13	17	21	40.730032	-73.990701	4	19
1	26	13	17	21	39	40.730032	-73.990701	4	33
2	13	17	21	39	27	40.730032	-73.990701	4	28
3	17	21	39	27	42	40.730032	-73.990701	4	37
4	21	39	27	42	53	40.730032	-73.990701	4	48

LINEAR REGRESSOR

Linear regressor:

Linear regression models the relationship between dependent and independent variables.
It assumes a linear relationship between the variables.
Widely used for prediction and inference in various domain

First Import the LinearRegression class from scikit-learn's linear_model module.
Fit the linear regression model to the training data using the fit method.
Predict the target variable for the testing data using the predict method and round the predicted values.
Predict the target variable for the training data using the predict method and round the predicted values.

```
from sklearn.linear_model import LinearRegression
lr_reg=LinearRegression().fit(df_train, tsne_train_output)

y_pred = lr_reg.predict(df_test)
lr_test_predictions = [round(value) for value in y_pred]
y_pred = lr_reg.predict(df_train)
lr_train_predictions = [round(value) for value in y_pred]
```

RANDOM FOREST

Random Forest:

Random forest is great for maintaining accuracy for a large proportion of data and doesn't allow overfitting if there are too many trees

```
RandomForestRegressor(max_features='sqrt', min_samples_leaf=4,
                      min_samples_split=3, n_estimators=40, n_jobs=-1)
```

```
y_pred = regr1.predict(df_test)
rndf_test_predictions = [round(value) for value in y_pred]
y_pred = regr1.predict(df_train)
rndf_train_predictions = [round(value) for value in y_pred]
```

```
#feature importances based on analysis using random forest
print (df_train.columns)
print (regr1.feature_importances_)
```

```
Index(['ft_5', 'ft_4', 'ft_3', 'ft_2', 'ft_1', 'lat', 'lon', 'weekday',
       'exp_avg'],
      dtype='object')
[0.00965372 0.03874404 0.11964895 0.12248967 0.32846239 0.00248174
 0.00283212 0.00174586 0.37394151]
```

XGBoost Regressor

XGBoost regressor:

Our next choice is XGBoost Regressor which is an implementation of gradient boosted decision trees designed for speed and performance. Training a hyper-parameter tuned Xg-Boost regressor on our train data.
The XGBoost Regressor is part of the XGBoost library, which is specifically designed for regression tasks. XGBoost offers a wide range of hyperparameters that control the model's behavior, including learning rate, number of estimators (trees), maximum tree depth etc.

As in the previous model, we calculated the Mean Absolute Percentage Error

```
{'ft_5': 1049.0, 'ft_4': 667.0, 'ft_3': 834.0, 'ft_2': 885.0,
'ft_1': 1060.0, 'lat': 715.0, 'lon': 793.0, 'weekday': 232.0,
'exp_avg': 706.0}
```

EVALUATION

```
Error Metric Matrix (Tree Based Regression Methods) -  MAPE
--------------------------------------------------------------------------------------------------
Baseline Model -                             Train:  0.1492063141406475       Test:  0.1541341143087541
Exponential Averages Forecasting -           Train:  0.14112388905282106       Test:  0.146041166537047
Linear Regression -                         Train:  0.14141633700523829      Test:  0.14682461624629067
Random Forest Regression -                   Train:  0.09712335885941091      Test:  0.14618996124926825
XgBoost Regression -                         Train:  0.13632742704186268       Test:  0.14492942373258
--------------------------------------------------------------------------------------------------
```

OTHER FEATURES

OTHER FEATURES - Weather

Weather conditions may impact the demand for taxi services in urban areas like NYC

- Use yearly data to better reflect the seasonal change in the weather
- Total number of ride records from 2015-2016: 277,171,036
- This dataset divides NYC into 263 different zones

	tpep_pickup_datetime	tpep_dropoff_datetime	PULocationID	DOLocationID
0	2015-01-01 00:11:33	2015-01-01 00:16:48	41	166
1	2015-01-01 00:18:24	2015-01-01 00:24:20	166	238
2	2015-01-01 00:26:19	2015-01-01 00:41:06	238	162
3	2015-01-01 00:45:26	2015-01-01 00:53:20	162	263
4	2015-01-01 00:59:21	2015-01-01 01:05:24	236	141
...	...	...	...	...

Weather dataset

Daily weather of NYC measured by LaGuardia Airport Station

Weather Metrics:

- Temperature
- Dew points
- Humidity
- Wind
- Pressure
- Precipitation

For each metric, maximum, minimum, and average is provided.

Date	Temperature_m	Temperature_av	Temperature_m	Dew_point_max	Dew_point_avg	Dew_point_min	Humidity_max
2015-01-01	38	33.3	28	18	10	5	46
2015-01-02	42	38.6	35	22	17.7	15	59
2015-01-03	41	35.5	32	40	29.5	19	100
2015-01-04	56	47.4	41	52	44.2	30	100
2015-01-05	49	35.3	22	28	12.9	3	52

Training and Testing dataset

- For cyclic columns such as time of day, day of year, encode with sine and cosine.
- For day of week, use one-hot encoding.
- 2015 data for training, 2016 data for testing.

LocationID	year	day_of_year	time_10min	pickup_count	date	is_holiday	day_of_week	temperature_max	temperature_avg	...
1	2015	1	0	0	2015-01-01	True	3	38	33.3	...
1	2015	1	1	0	2015-01-01	True	3	38	33.3	...
1	2015	1	2	0	2015-01-01	True	3	38	33.3	...
1	2015	1	3	0	2015-01-01	True	3	38	33.3	...
1	2015	1	4	0	2015-01-01	True	3	38	33.3	...

Ensemble Models

Random Forest:
Mean Squared Error: 47.81
Mean Absolute Percentage Error: 2.25

Gradient Boosting:
Mean Squared Error: 464.62
Mean Absolute Percentage Error: 12.27

Ada Boost:
Mean Squared Error: 1237.62
Mean Absolute Percentage Error: 30.72

Linear Models

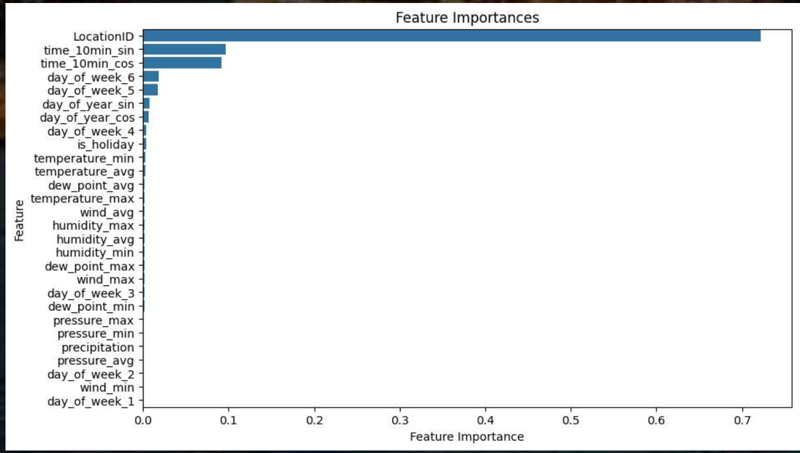
Linear Regression:
Mean Squared Error: 625.53
Mean Absolute Percentage Error: 14.96

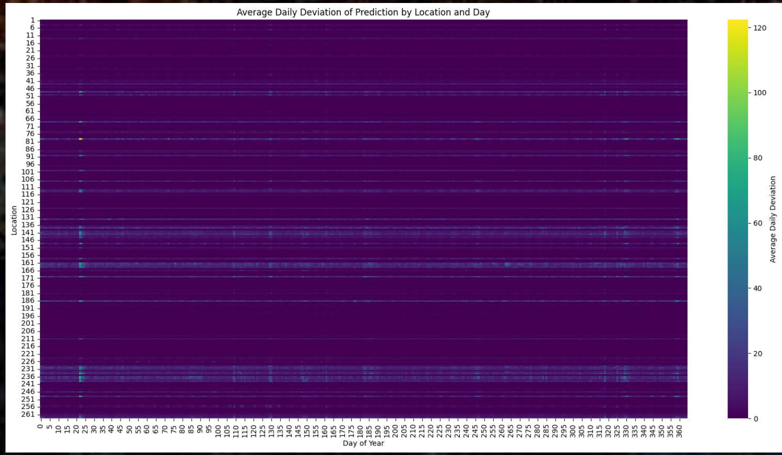
Ridge:
Mean Squared Error: 625.53
Mean Absolute Percentage Error: 14.96

Bayesian Regression:
Mean Squared Error: 625.47
Mean Absolute Percentage Error: 14.96

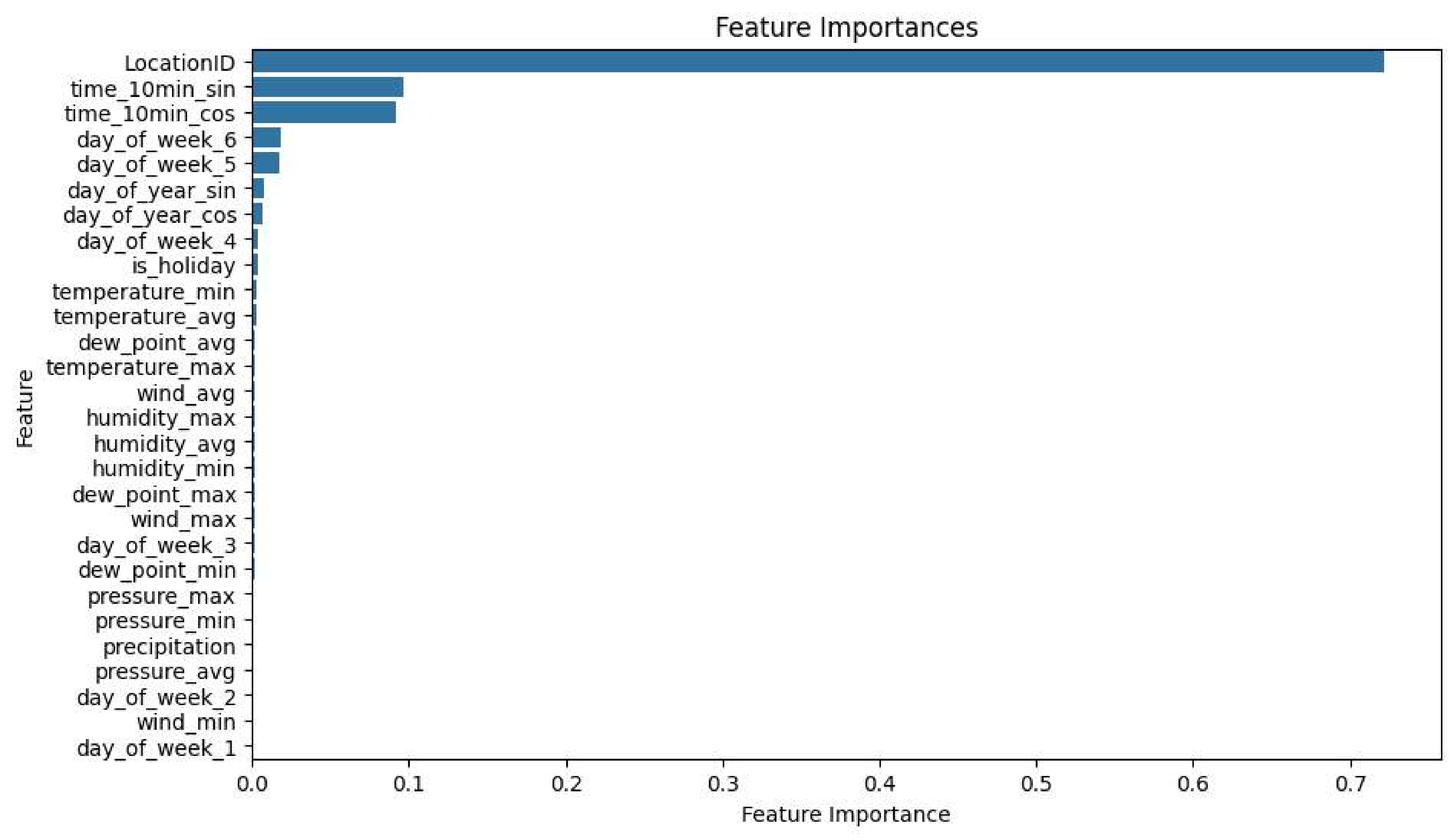
Random Forest Regressor

- Estimator: 16
- Mean Squared Error: 47.81
- Mean Absolute Percentage Error: 2.25

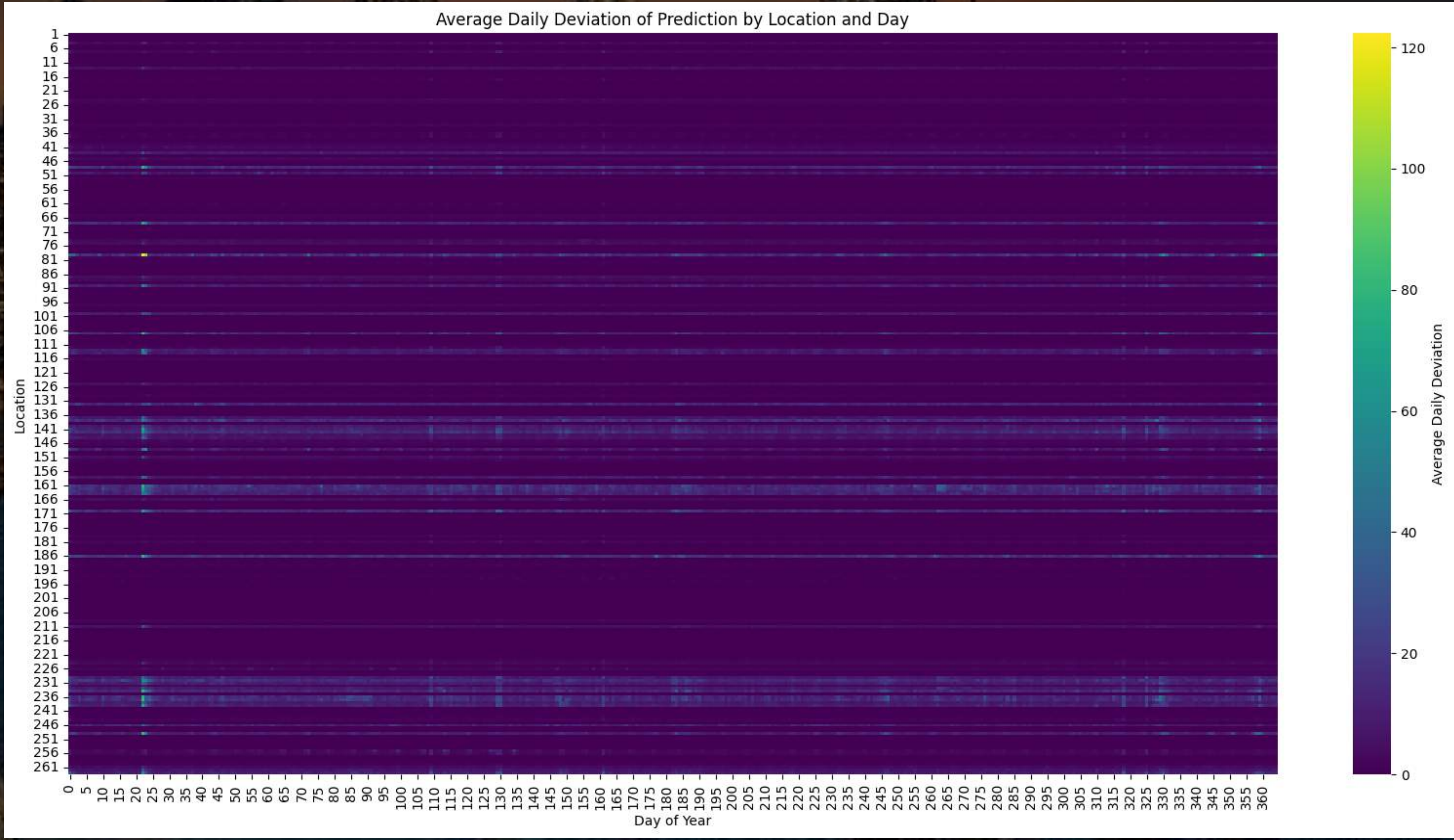




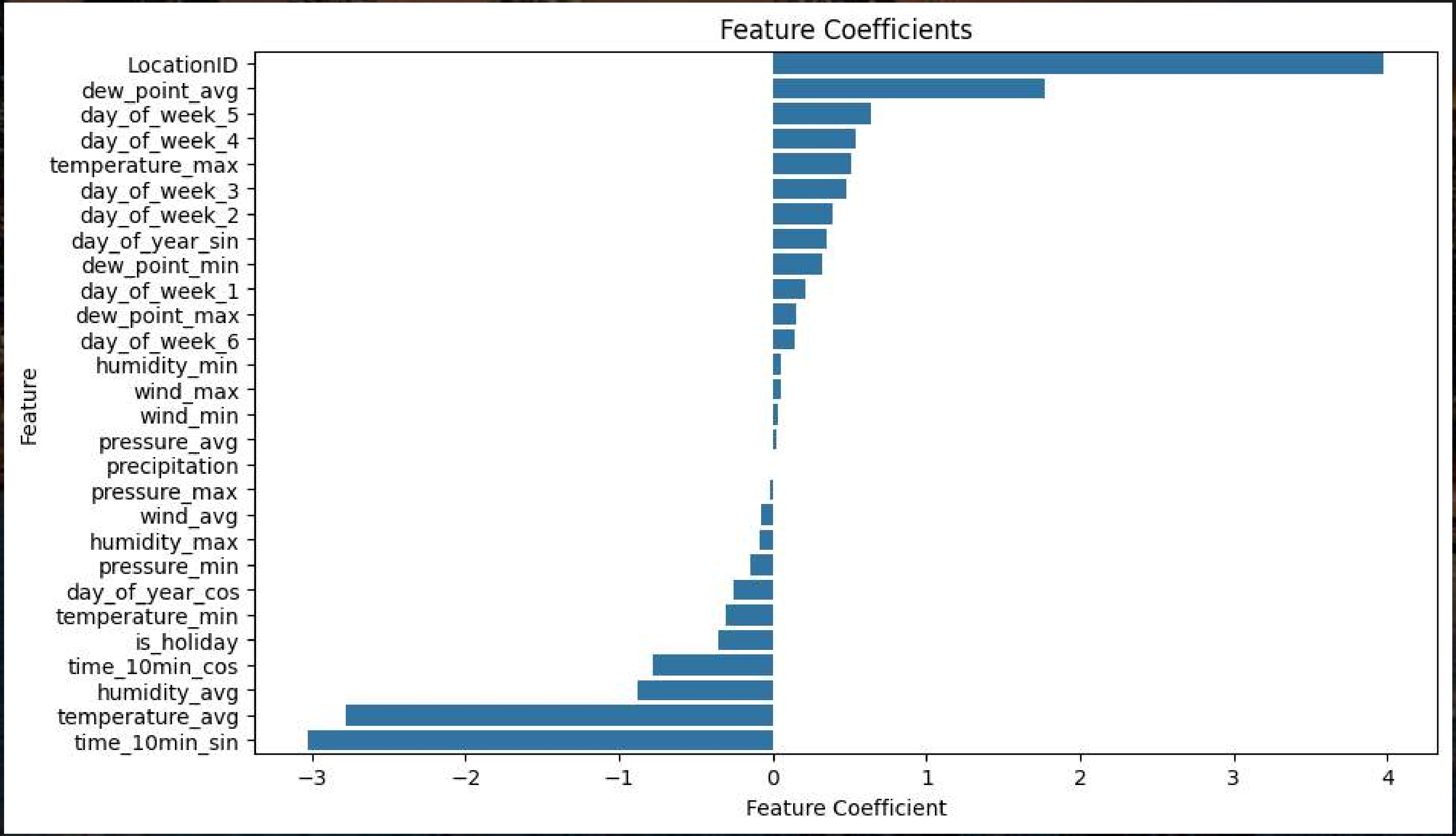
Random Forest Regressor



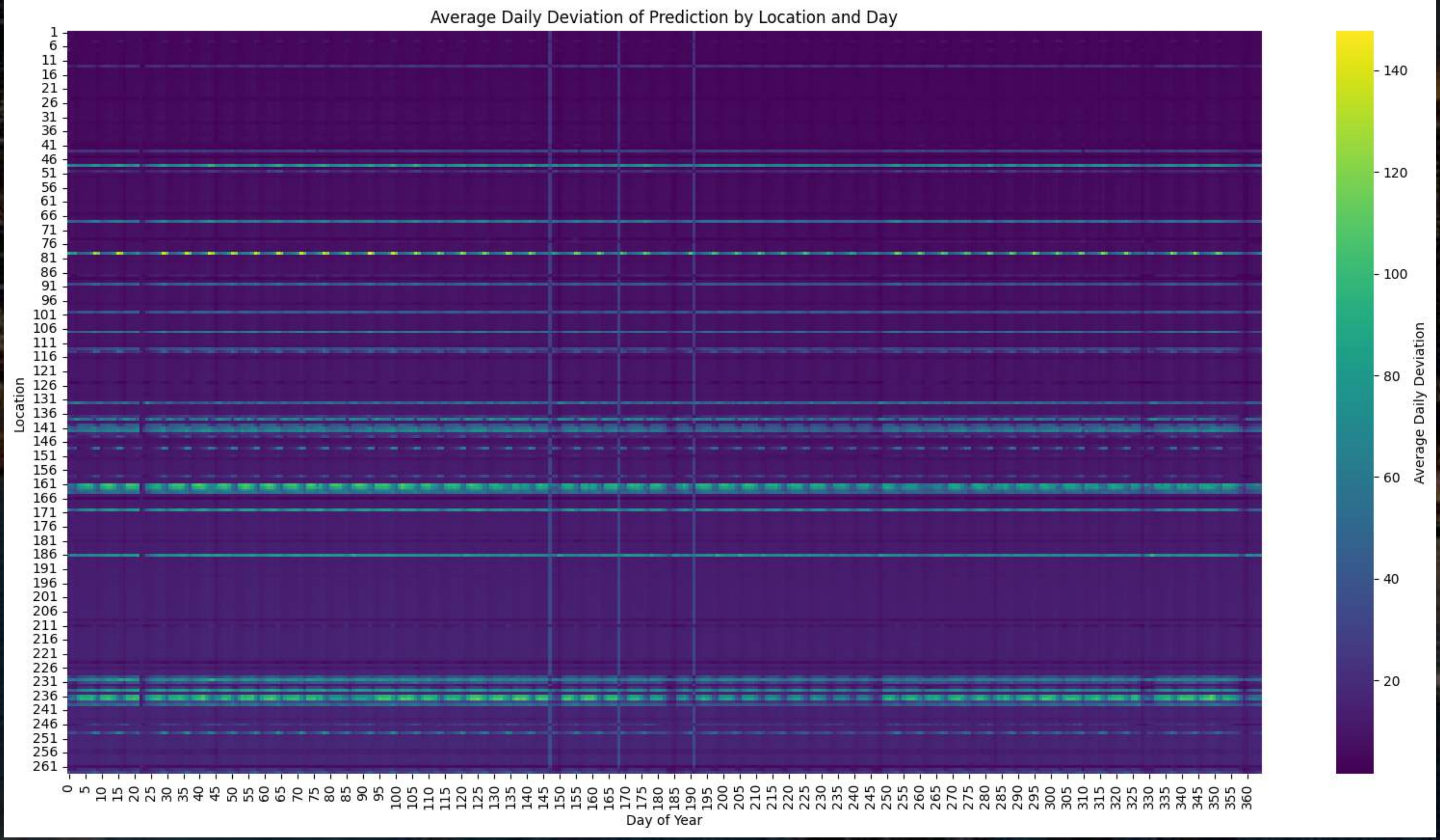
Random Forest Regressor



Linear Regression Models



Linear Regression Models



FUTURE IMPROVEMENTS

Future Improvement

Performance Issue

- Processed dataset takes 4.5GB in RAM (2 years of data)
- For 64GB RAM PC, training crashes if random forest regressor has over 16 estimators

Proposed solutions

- Implement incremental training for the model and feed data in chunks
- Train small models for each location separately

Other features

- Economics

Q&A