
CabEdge Solutions
A Driver First Initiative

01 **What does CabEdge Offer?**

02 **Exploratory Data Analysis**

03 **Technical Analysis**

04 **Non Technical Analysis**

INTRODUCTION

CabEdge

An one stop solution for all the drivers .

Stay Ahead and Stay strong with CabEdge..!!

01

What does CabEdge offers?

What we do?

We recommend the ‘Zone’ the Taxi Driver should drive to after dropping off a passenger to receive the fastest next booking depending on the time/day of the week/month and weather conditions.

This reduces the “Waiting time between Bookings” for the drivers and increases earning potential for both the Driver and the company.

CabEdge **Solutions**

PROBLEM

A Taxi or a Rideshare driver earns an average of 15 -25 USD per hour.

Rideshare driver pay averages only $9.21 an hour after deductions for Rideshare fees, vehicle expenses, payroll taxes, and the cost of a “modest benefits package.”

Cost of a Taxi license plate in 2013 in New York : $1,000,000

Airport Taxi Queue in New York City

LA Guardia Airport NYC

JFK Airport Terminal 7 NYC

EWR Newark Airport

Airport Driver Queues: Anywhere from 5 minutes to 3 hours. Depends on when, size of airport, supply, etc.

SOLUTION

The biggest issue a driver faces is the waiting time for their next booking.

Every minute the driver spends on the road without a booking is a loss for both the driver and the company.

CabEdge precisely recommends them the area they can drive to for their next fastest booking based on their current location and current weather conditions

Potential Audiences?

Drivers

As CabEdge solution is open source, it can be used by any driver irrespective of the company he is driving for.

Note : The solution is based on NYC Yellow cabs data set.

Yellow Taxi ,Silver Cabs, UBER, OLA, LYFT, BOLT

Any Taxi or rideshare company can collaborate with us to suit their platform and benefit their driver with our solution

Riders

Reduces the waiting time for the Riders and the solution marks the Hot zones at a specific hour of the day

Time for a Joy Ride

CabEdge Solutions

Click here for a live Demo of our Solutions.

https://cabedge.herokuapp.com

Drive Safe .. Drive Smart..!!

02

Exploratory Data Analysis

Data Dictionary

Field Name	Description
VendorID	A code indicating the TPEP provider that provided the record. **1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.**
tpep_pickup_datetime	The date and time when the meter was engaged.
tpep_dropoff_datetime	The date and time when the meter was disengaged.
Passenger_count	The number of passengers in the vehicle. This is a driver-entered value.
Trip_distance	The elapsed trip distance in miles reported by the taximeter.
PULocationID	TLC Taxi Zone in which the taximeter was engaged
DOLocationID	TLC Taxi Zone in which the taximeter was disengaged
RateCodeID	The final rate code in effect at the end of the trip. **1= Standard rate** **2=JFK** **3=Newark** **4=Nassau or Westchester** **5=Negotiated fare** **6=Group ride**
Store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka "store and forward," because the vehicle did not have a connection to the server. **Y= store and forward trip** **N= not a store and forward trip**
Payment_type	A numeric code signifying how the passenger paid for the trip. **1= Credit card** **2= Cash** **3= No charge** **4= Dispute** **5= Unknown** **6= Voided trip**
Fare_amount	The time-and-distance fare calculated by the meter.
Extra	Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
MTA_tax	$0.50 MTA tax that is automatically triggered based on the metered rate in use.
Improvement_surcharge	$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
Tip_amount	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
Tolls_amount	Total amount of all tolls paid in trip.
Total_amount	The total amount charged to passengers. Does not include cash tips.

Important Predictors

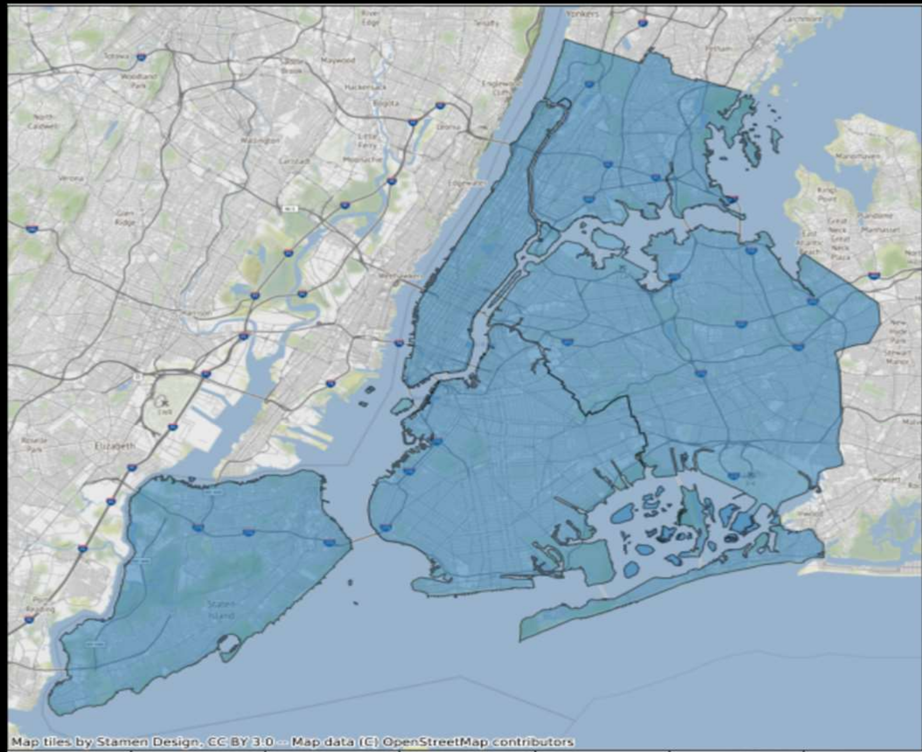
- Tpep_pickup_datetime
- Tpep_dropoff_datetime
- PULocationID
- DOLocationID
- Fare_amount

8.200.000

Average number of Yellow Cabs trips per month in 2018 in New York City

New York City Boroughs

- Bronx
- Manhattan
- Brooklyn
- Queens
- Staten Island

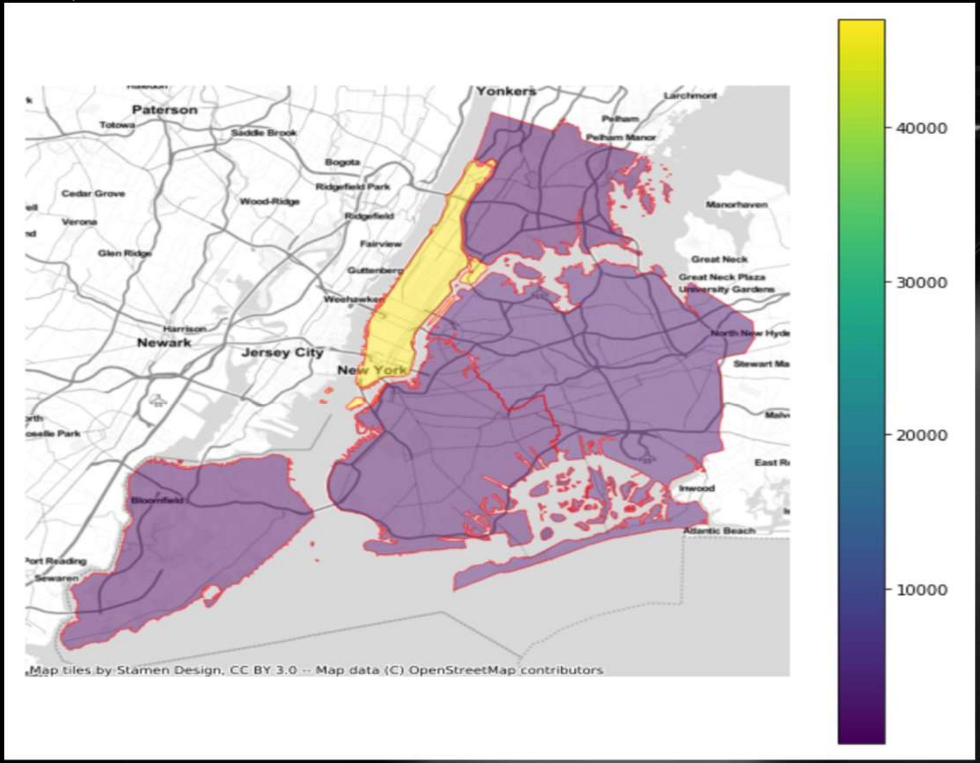


New York City Zones

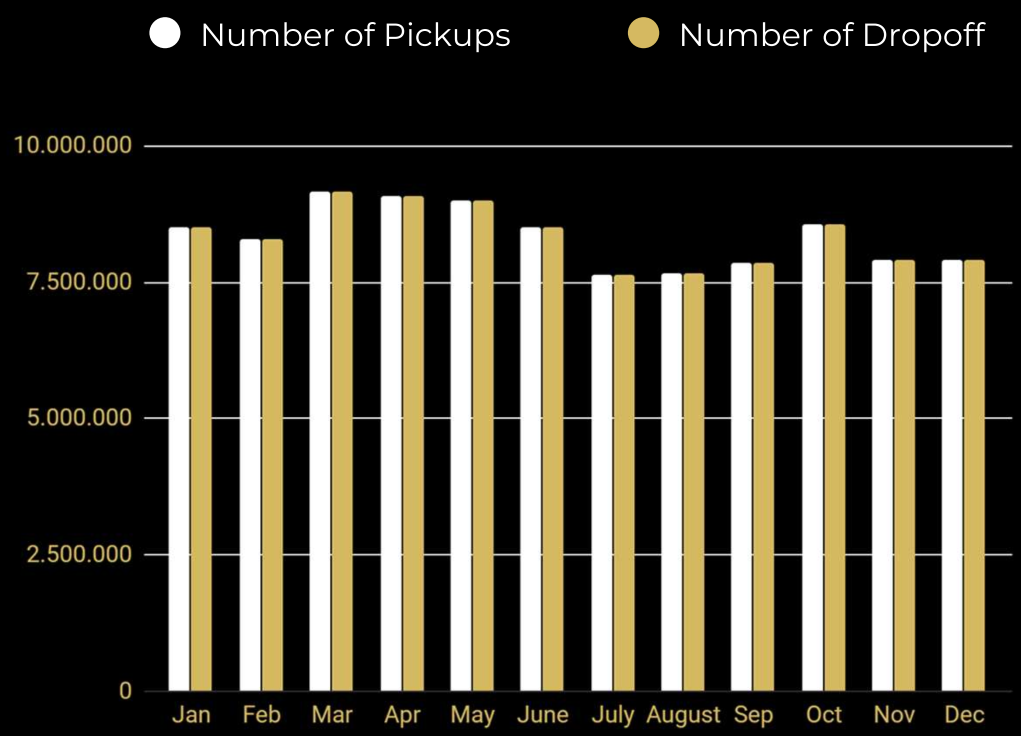
- Bronx - 43 zones
- Manhattan - 69 zones
- Brooklyn - 61 zones
- Queens - 69 zones
- Staten Island - 20 zones

Total Zones - 263

Number of Pickups at 1 pm to 2 pm on 7th Jan, 2019



Monthly Trends (Pickup vs Dropoff)



Mar, Apr, May

Busiest Months

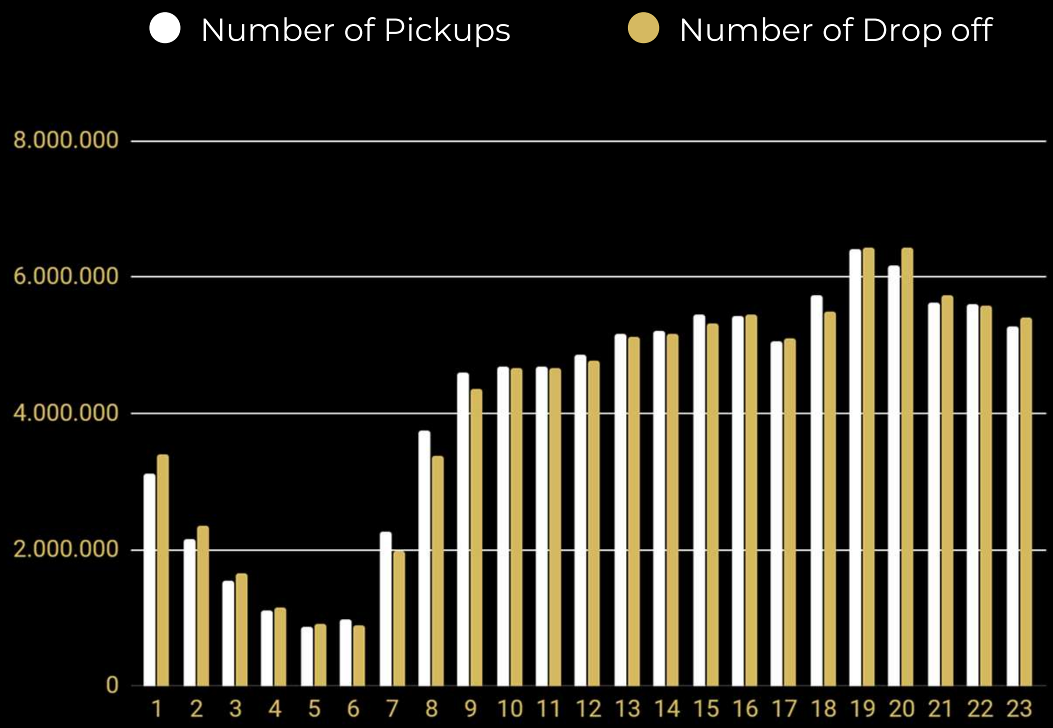
10,00,57,91
3

Total Trips in 2018

2,74,131

Trips per day

Hourly Trends (Pickup vs Drop off)



6 pm to 9 pm
Peak Hours

Busiest Hours

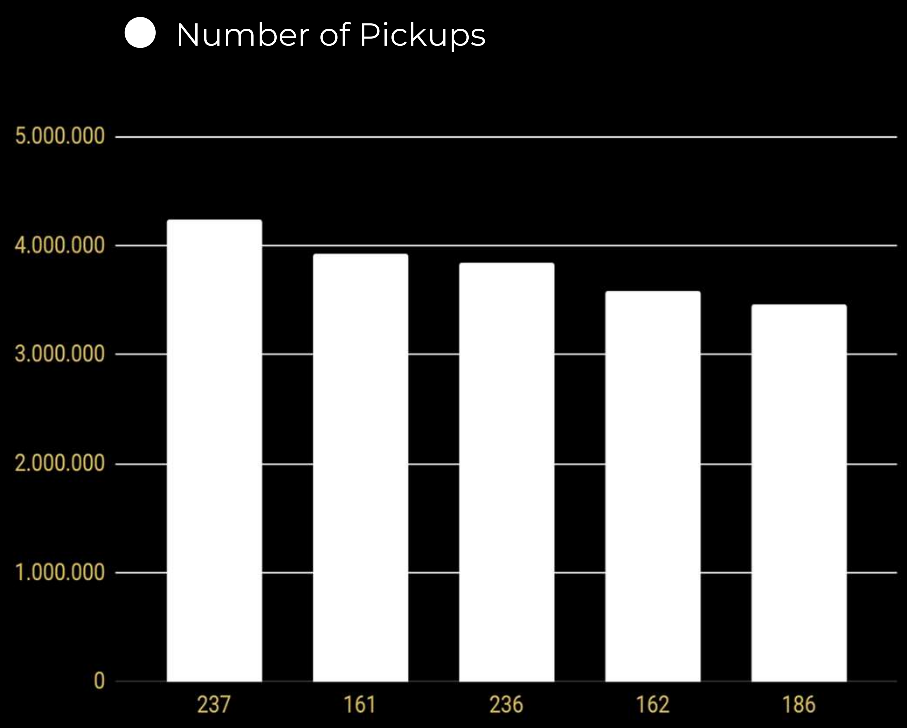
10,00,57,91
3

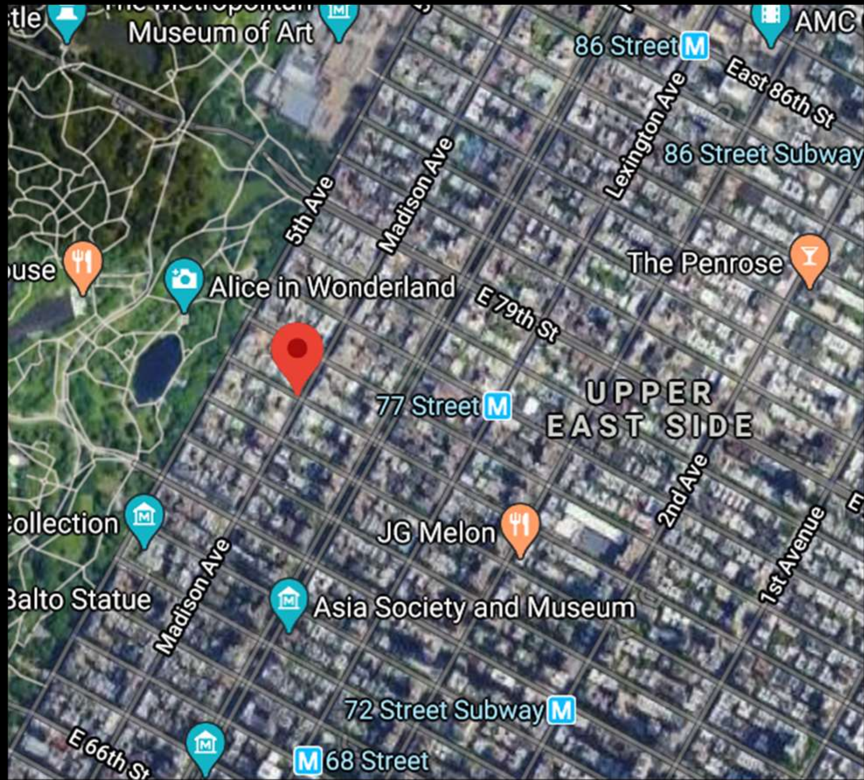
Total Trips in 2018

11,422

Average Trips per hour

Busiest PickUp Zones

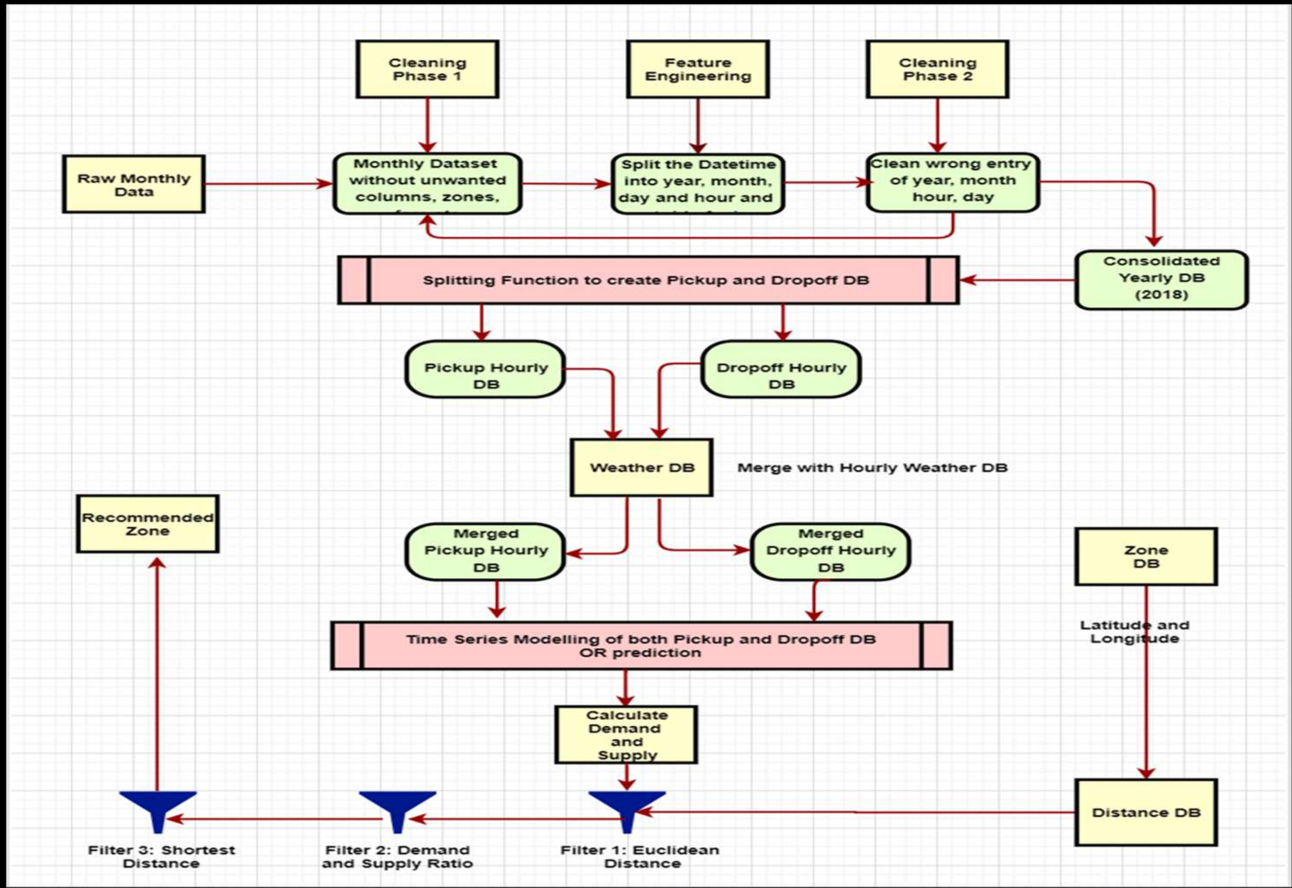




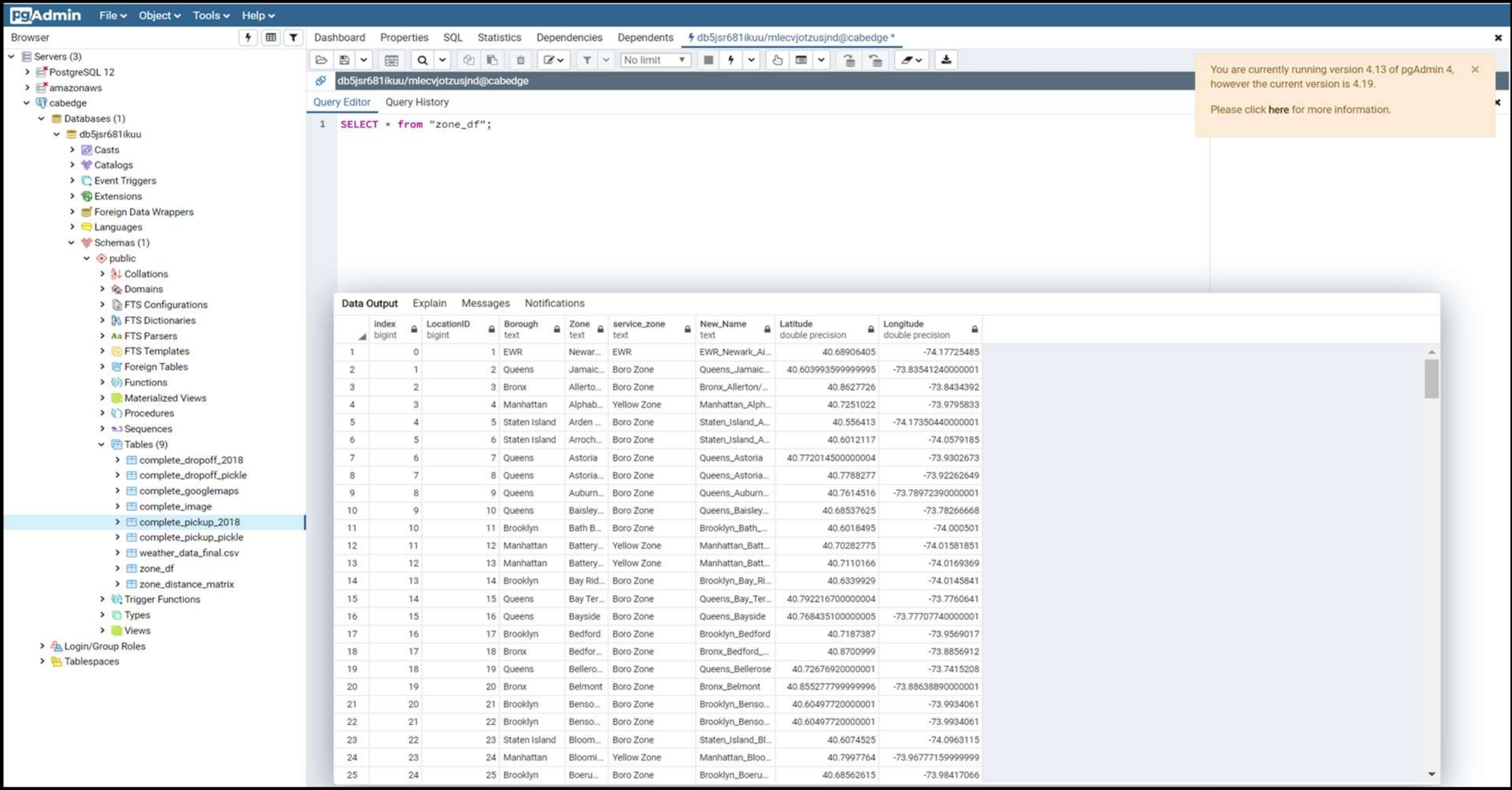
03

Technical Analysis

CabEdge Flow Chart/Diagram



Example of a Database



CabEdge Raw 2018 Monthly DB

```
df.head()
```

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID
0	1	2018-01-01 00:21:05	2018-01-01 00:24:23	1	0.5	1	N	41	24
1	1	2018-01-01 00:44:55	2018-01-01 01:03:05	1	2.7	1	N	239	140
2	1	2018-01-01 00:08:26	2018-01-01 00:14:21	2	0.8	1	N	262	141
3	1	2018-01-01 00:20:22	2018-01-01 00:52:51	1	10.2	1	N	140	257
4	1	2018-01-01 00:09:18	2018-01-01 00:27:06	2	2.5	1	N	246	239

```
df.tail()
```

re_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount
N	158	163	1	12.0	0.5	0.5	2.65	0.0	0.3	15.95
N	163	162	1	4.5	0.5	0.5	1.15	0.0	0.3	6.95
N	74	69	2	10.5	0.5	0.5	0.00	0.0	0.3	11.80
N	7	193	2	0.0	0.0	0.0	0.00	0.0	0.0	0.00
N	7	193	2	0.0	0.0	0.0	0.00	0.0	0.0	0.00

1. Feature Engineering and Data Wrangling

   a. Phase 1 cleaning (Unwanted columns, wrong zones, fares etc)

   b. Feature Engineering to split the timestamp into year,day, month , hour and get rid of minutes and seconds.

   c. Phase 2 cleaning to clean wrong entry of dates

CabEdge Complete Pickup DB

	datetime	apparentTemperature	windSpeed	PULocationID	pickup_year	pickup_month	pickup_day	pickup_hour	Count
0	2018-04-01 00:00:00	40.42	3.27	1.0	2018	4	1	0	0.0
1	2018-04-01 01:00:00	45.16	2.28	1.0	2018	4	1	1	0.0
2	2018-04-01 02:00:00	48.38	1.34	1.0	2018	4	1	2	0.0
3	2018-04-01 02:00:00	51.37	1.68	1.0	2018	4	1	2	0.0
4	2018-04-01 03:00:00	53.91	2.96	1.0	2018	4	1	3	0.0

```
df.tail()
```

	datetime	apparentTemperature	windSpeed	PULocationID	pickup_year	pickup_month	pickup_day	pickup_hour	Count
2303875	2018-09-30 19:00:00	54.16	4.43	263.0	2018	9	30	19	279.0
2303876	2018-09-30 20:00:00	53.53	5.02	263.0	2018	9	30	20	208.0
2303877	2018-09-30 21:00:00	52.96	4.77	263.0	2018	9	30	21	171.0
2303878	2018-09-30 22:00:00	55.63	5.04	263.0	2018	9	30	22	135.0
2303879	2018-09-30 23:00:00	57.95	5.36	263.0	2018	9	30	23	65.0

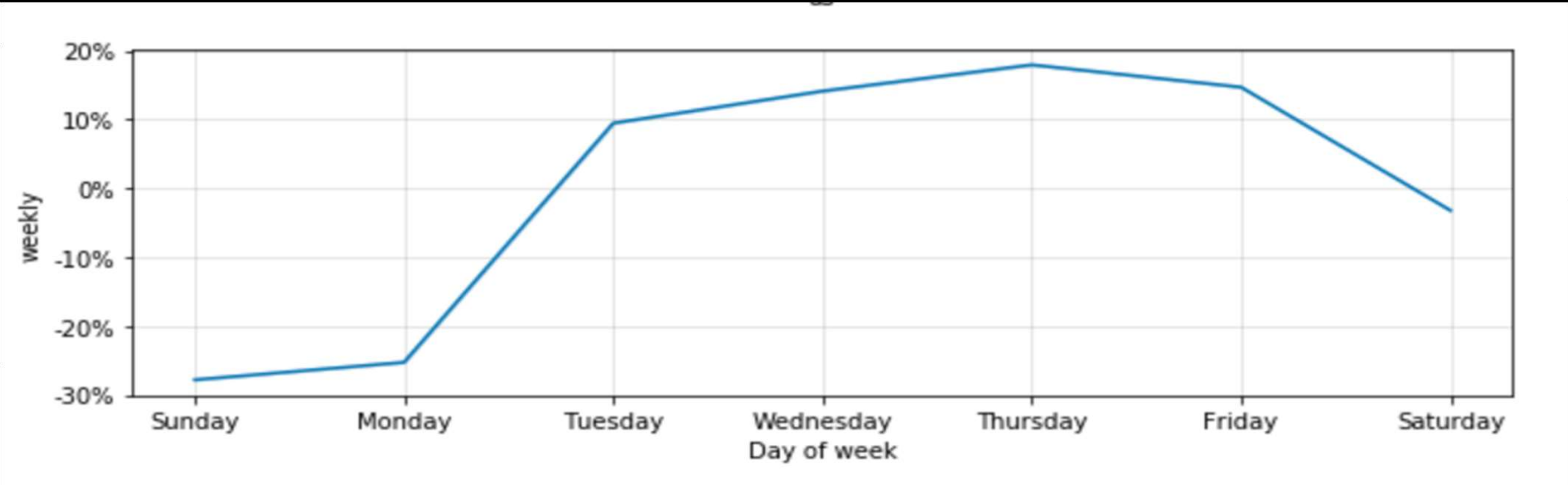
263 zones x 365 days x 24 hours = **2,303,880** Rows

2. FBProphet Model Creation

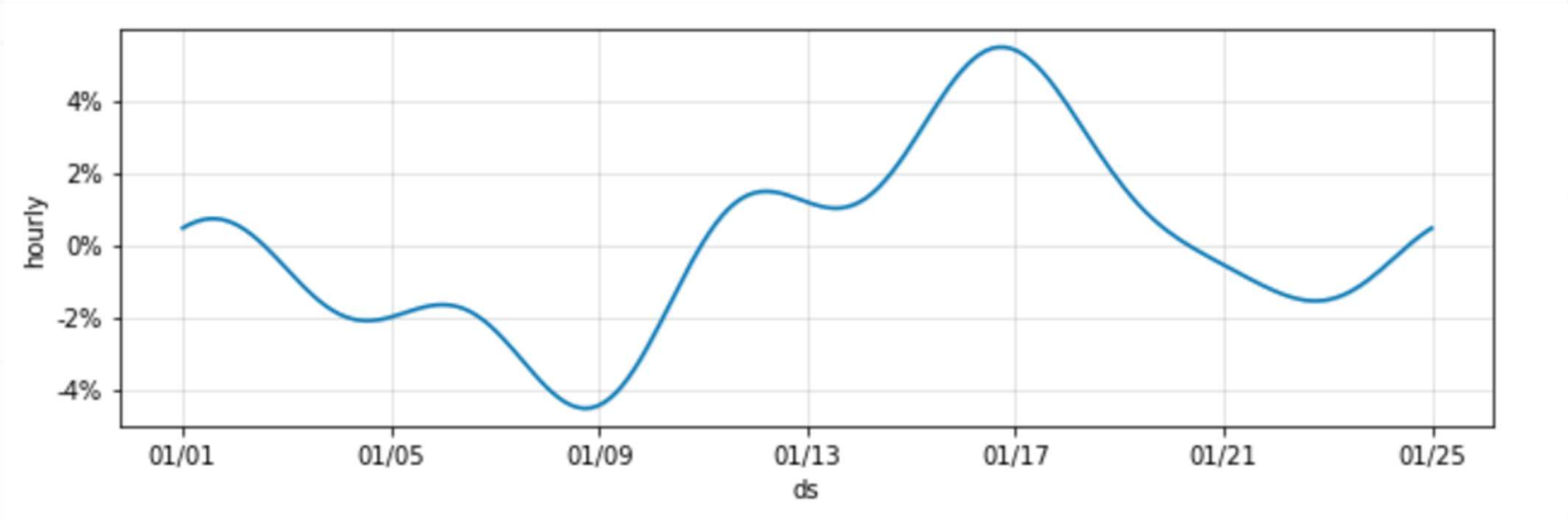
```
m1 =
Prophet(growth='logistic',changepoint_prior_scale=35,seasonality_prior_scale=30,holidays_prior_scale=20,seasonality_mode='multiplicative',yearly_seasonality=False, weekly_seasonality=False,daily_seasonality=False)

m1.add_country_holidays(country_name='US')

m1.add_seasonality(name='yearly', period=365,
fourier_order=12,mode='multiplicative')

m1.add_seasonality(name='monthly', period=30.5,
fourier_order=35,mode='multiplicative')

m1.add_seasonality(name='weekly', period=7,
fourier_order=40,mode='multiplicative')

m1.add_seasonality(name='daily', period=1,
fourier_order=30,mode='multiplicative')

m1.add_seasonality(name='hourly', period=24,
fourier_order=20,mode='multiplicative')

m1.add_regressor('apparentTemperature')

m1.add_regressor('windSpeed')
```

Yearly Trend

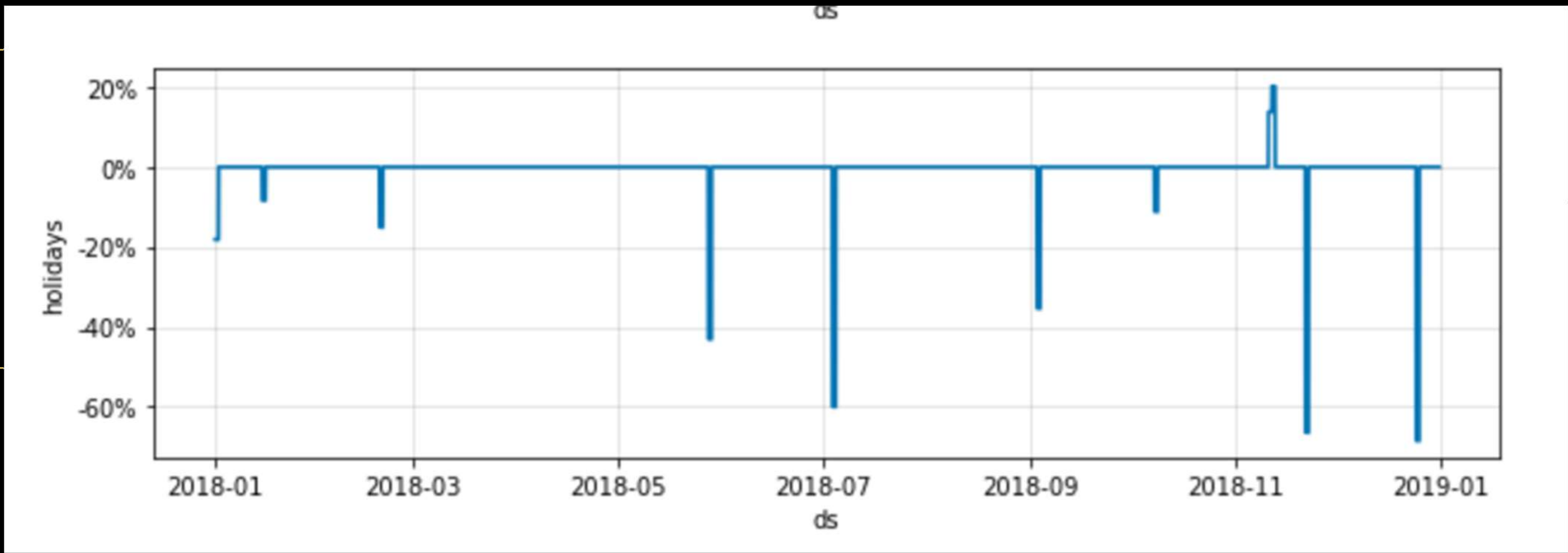
Weekly Seasonality



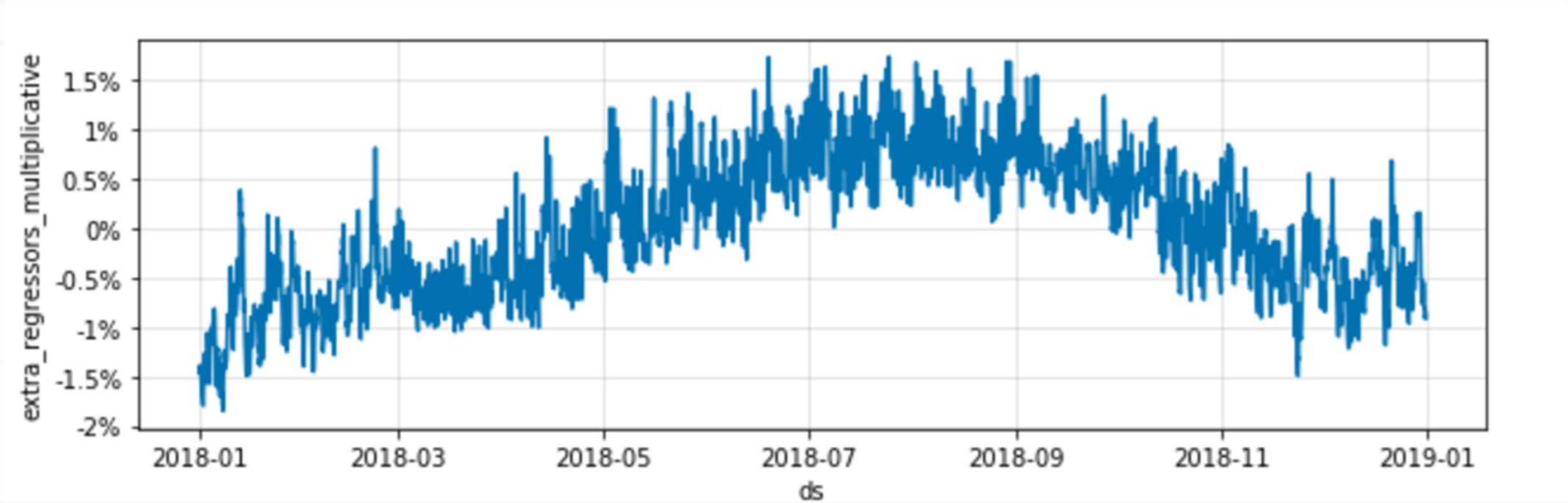
Daily Seasonality



Holiday Seasonality



Weather Seasonality(Temperature and Wind Speed)



Error Calculation Metrics

Actual Nb of Pickup - 10

MAE : 90

Predicted Nb of Pickup - 100

Actual Nb of Dropoff - 20

MAE : 90

Predicted Nb of Drop off - 200

Actual Ratio - 10 : 20 - 0.5

MAE : 0

Predicted Ratio - 100:200 -0.5

“Hence it is the ‘Ratio’ that is more relevant to the problem description”

Quality Metrics (MAE, MSE, RMSE)

1. Demand and Supply Ratio = Nb of pickup/ Nb of Drop off
2. Determine the Baseline Score of the 'Ratios' through Persistence Algorithm (B_MSE, B_RMSE)
3. Calculate MAE, MSE and RMSE of the 'Ratios' for all zones for length of time period similar to Baseline
4. Compare the RMSE.

```
[1139]: combined_ratio_score.head(10)
```

[1139]:

	MAE	MSE	RMSE	zone	B_MSE	B_RMSE	result	percentage
0	0.009349	0.002575	0.050742	1	0.003955	0.062890	low	-19.316384
1	0.004032	0.004032	0.063500	2	0.003859	0.062121	high	2.219410
2	0.100468	0.126464	0.355618	3	0.154896	0.393568	low	-9.642613
3	0.233101	0.144496	0.380126	4	0.210700	0.459021	low	-17.187633
4	0.000000	0.000000	0.000000	5	0.002013	0.044871	low	-100.000000
5	0.024866	0.024530	0.156619	6	0.059923	0.244791	low	-36.019221
6	0.286517	0.205055	0.452829	7	0.229662	0.479231	low	-5.509106
7	0.009856	0.012246	0.110661	8	0.050285	0.224242	low	-50.651258
8	0.074595	0.077508	0.278403	9	0.114592	0.338515	low	-17.757635
9	0.745966	1.880566	1.371337	10	2.952685	1.718338	low	-20.193950

Faulty Zones (MAE, MSE, RMSE)

Does it affect the Recommendation? 3.3%

1. Demand and Supply Ratio Filter = Nb of pickup/ Nb of Drop off > 1
2. For All the Faulty zones (B_RMSE < RMSE) , 3.3% of ratio calculations can be faulty.
3. Most of this zones will naturally be filtered out in the 2nd layer of Filtering Process

```
new_dataframe = pd.DataFrame()
for zone in range(1,264):
    if zone in faulty_zones_month_list:
        temp_predicted_ratio_df = jan_predicted_ratio_df[jan_predicted_ratio_df['zone'] == faulty_zone]
        temp_predicted_ratio_df['faulty'] = 'yes'
        temp_predicted_ratio_df['DSR'] = np.where((temp_predicted_ratio_df['actual_ratio'] > 1) & (temp_predicted_ratio_df['predicted_ratio'] < 1),
                                          'high', 'low')
    else:
        temp_predicted_ratio_df = jan_predicted_ratio_df[jan_predicted_ratio_df['zone'] == zone]
        temp_predicted_ratio_df['faulty'] = 'No'
        temp_predicted_ratio_df['DSR'] = 'Does not matter'

    new_dataframe = new_dataframe.append(temp_predicted_ratio_df)
```

```
new_dataframe['DSR'].value_counts()
```

```
Does not matter    122760
low                 66346
high                 6566
Name: DSR, dtype: int64
```

Margin of Error(MOE)

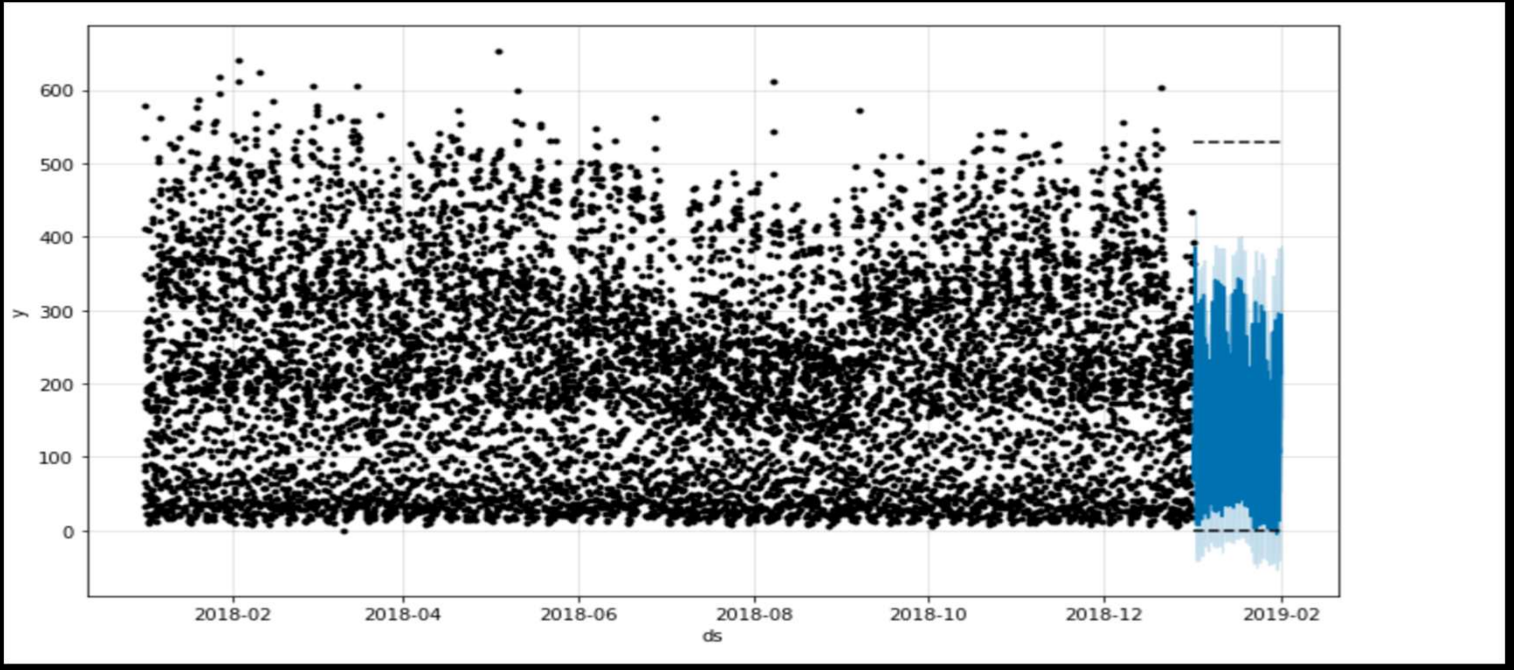
How long can we predict ?

1. As the growth is ‘Logistic’, the MOE is abrupt.
2. MOE = (predicted_value - upperband)/predicted_value * 100

```
new_dataframe[new_dataframe['zone'] == 263].head()
```

e	predicted_pickup_count	actual_pickup_count	p_yhat_lower	p_yhat_upper	actual_dropoff_count	predicted_dropoff_count	d_yhat_lower	d_yhat_upper	actual_ratio	predicted_ratio	lower_band	upper_band	MOE
.0	100	375.0	31	167	320.0	153	77	227	1.172	0.654	0.403	0.736	12.538226
.0	77	463.0	4	143	484.0	112	40	187	0.957	0.688	0.100	0.765	11.191860
.0	53	367.0	0	121	435.0	77	3	149	0.844	0.688	0.000	0.812	18.023256
.0	36	250.0	0	110	295.0	51	0	127	0.847	0.706	0.000	0.866	22.662890
.0	42	128.0	0	121	127.0	39	0	113	1.008	1.077	0.000	1.071	-0.557103

Margin of Error(MOE) Interpretation



The Model can predict to +/- X % range at that particular time of the day.

04

Non Technical Analysis

Welcome to

CabEdge Solutions

Click on the below link to check a live Demo of our Solutions.


Drive
Without
worry
with
CabEdge

THANKS!

Do you have any questions?
arnabm.au@gmail.com
+61 413 611 312

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Please keep this slide for attribution.