

# DATA PROCESSING ON A BIG DATASET WITH DATABRICKS SPARK

Simon Lim

Big Data Engineering

# 1 CONTEXTS AND OBJECTIVES OF THE PROJECT

---

The New York City Taxi and Limousine Commission (TLC) is the agency responsible for regulating and managing New York City's taxi cabs. TLC regularly records millions of trips information from both yellow and green taxi cabs, including pick-up and drop-off dates/times, locations, trip distances, payment types and total amount etc. While yellow taxi cabs are the iconic taxi vehicles, which can pick up passengers anywhere in the city, green taxi cabs can only pick up passengers in certain designated areas.

In this regard, the objective of the project is to analyse a large dataset and obtain meaningful outcomes using Spark in Databricks. The procedure of the project includes data ingestion and preparation, data transformation and analysis and finally training a Machine Learning algorithm for predicting total amount of trips. In this project, Databricks is a main platform used to perform data preparation, cleaning and machine learning.

## 2 PRESENTATION OF DATASET & PREDICTIVE ISSUES

---

There are yellow and green taxi cabs datasets from 2015 to 2022 for each taxicab and they are all parquet files. Each dataset contains various information of trips, including VendorID, pickup\_datetime, dropoff\_datetime, LocationID, passenger\_count, trip\_distance, fare\_amount and total\_amount etc. Also, there is a location dataset, including LocationID, Borough, zone and service\_zone.

Each 2015-2022 dataset for yellow and green taxi contains a massive amount of data and therefore they have been compressed by saving them as parquet files. The total number of yellow taxi trips is 663055251 and the total number of green taxi trips is 66200401. At the beginning, it was assumed that a massive amount of data would considerably make a delay for the processing of data cleaning, analysis and machine learning action phases. Hence, the process of saving and re-loading dataframes would be essential to reduce the significant processing delay.

In terms of datasets, parquet files are mainly used in this project, as they compress data and hence be able to include a more massive amount of data.

### 3 PIPELINE OF THE PROJECT

---


0. Before ingesting data, there are two necessary processes to be set up. The first step was to set up a cloud storage account on Microsoft Azure and Databricks. Subsequently, it was also necessary to mount an Azure Blob storage container to Databricks file system.

#### PART 1. Data Ingestion and Preparation

1. After mounting Azure storage container, all parquet files in Azure container were shifted to Databricks file system.
2. On Databricks, multiple parquet files of yellow taxi trips from 2015 to 2022 were combined and loaded as a data frame. Parquet files of green taxi trips were loaded in same way yellow taxi trips data did.
3. The total number of yellow taxi trips was 663055251 and the total number of green taxi trips was 66200401.
4. A dataset containing only green taxi trips in 2015 was exported as a csv file to Azure container and compared to a parquet file of 2015 green trips.

 	part-00000-tid-222718995982429228-38226f47-...	25/09/2023, 4:21:26 ...	Hot (Inferred)	Block blob	685.02 MiB
 	part-00001-tid-222718995982429228-38226f47-...	25/09/2023, 4:21:10 ...	Hot (Inferred)	Block blob	682.83 MiB
 	part-00002-tid-222718995982429228-38226f47-...	25/09/2023, 4:20:16 ...	Hot (Inferred)	Block blob	710.31 MiB

Figure 1. Green\_taxi\_2015.csv version

 	green_taxi_2015.parquet	21/09/2023, 11:27:02...	Hot (Inferred)	Block blob	385.81 MiB
 	green_taxi_2016.parquet	21/09/2023, 11:25:04...	Hot (Inferred)	Block blob	330.67 MiB
 	green_taxi_2017.parquet	21/09/2023, 11:21:50...	Hot (Inferred)	Block blob	239.85 MiB

## Figure 2. Green\_taxi\_2015.parquet version

As two figures show, the size of parquet file is 385.81 MiB, while the size of csv file is 685.02 + 682.83 + 710.31 MiB. **Given that the amount of data in this project is massive, parquet version is more appropriate than csv version in this project.**

5. Before data cleaning phase, there were some changes in columns for yellow and green trip datasets (also shown in Figure 3).
  - First, columns for pickup and dropoff datetiems were re-named as “pickup\_datetime” and “dropoff\_datetime” for both datasets, in order to combine two datasets in later stage.
  - Trip distance in miles was converted to km.
  - Three new columns were created, including trip durations in seconds, trip speed in km/h and taxi color (yellow or green).

```
date_format_pattern = "yyyy-MM-dd HH:mm:ss" # Setting a variable for date format pattern, including year, month, date, hour, minutes and seconds

df_yellow = df_yellow.withColumnRenamed("tpep_pickup_datetime", "pickup_datetime").\
    withColumnRenamed("tpep_dropoff_datetime", "dropoff_datetime").\
    withColumn("trip_distance_km", col("trip_distance") * 1.60934).\
    withColumn("trip_duration_sec", expr("unix_timestamp(dropoff_datetime) - unix_timestamp(pickup_datetime)").\
    withColumn("speed_kmh", expr("trip_distance_km / (trip_duration_sec / 3600)").\
    withColumn("taxi_colour", lit("yellow"))

#changing the name of column for pickup
#changing the name of column for dropoff
#converting from miles to km for distance
#creating a column for trip duration in seconds.
#creating a column for trip speed in km/h
#creating a column for taxi colour

display(df_yellow)
```

**Figure 3. Adding new columns, renaming columns and changing column unit.**

6. In cleaning phase, yellow and green dataset was filtered with an appropriate range of date, speed, distance and duration of trips. Thus, all data outside of the range were removed.

```

start_time= "2015-01-01 00:00:00" # The earliest date of trips
end_time = "2022-12-31 23:59:59" # The latest date of trips
min_speed_kmh = 0 # Non-negative speed
max_speed_kmh = 40.2336 # Maximum speed in NYC citywide
min_duration_sec = 60 # Minimum duration of trip
max_duration_sec = 3000 # Maximum duration of trip
min_distance_km = 0.5 # Minimum distance of trip
max_distance_km = 18 # Maximum distance of trip

df_yellow_cleaned = df_yellow.filter(expr("unix_timestamp(dropoff_datetime) > unix_timestamp(pickup_datetime)")
    (col("pickup_datetime").between(start_time, end_time)) &
    (col("dropoff_datetime").between(start_time, end_time)) &
    (col("speed_kmh").between(min_speed_kmh, max_speed_kmh)) &
    (col("trip_duration_sec").between(min_duration_sec, max_duration_sec)) &
    (col("trip_distance_km").between(min_distance_km, max_distance_km)) &
    (col("passenger_count").between(1, 5))
)

```

**Figure 4. Cleaning data**

7. Before combining yellow and green dataframes, they were saved as parquet files and re-loaded because the process of saving and re-loading can potentially reduce the times taken for processing of Spark actions.
8. Cleaned yellow and green datasets were then combined using union. In order to combine two dataframes, common columns were selected, including “trip\_distance\_km”, “trip\_duration”, “speed\_kmh”, “pickup\_datetime”, “droff\_datetime”, “passenger\_count”, “PULocationID”, “DOLocationID”, “total\_amount”, “tip\_amount”, “tolls\_amount” and “taxi\_colour”.
9. The combined dataframe was also saved and re-loaded.
10. Location data was then loaded and then split into two dataframes, pickup\_location and dropoff\_location. Additionally, “LocationID” and “Borough” columns were created for each pickup\_location and dropoff\_location.

display(location\_df)

► (1) Spark Jobs

Table ▾ +

	LocationID ▲	Borough ▲	Zone ▲	service_zone ▲
1	1	EWB	Newark Airport	EWB
2	2	Queens	Jamaica Bay	Boro Zone
3	3	Bronx	Allerton/Pelham Gardens	Boro Zone
4	4	Manhattan	Alphabet City	Yellow Zone
5	5	Staten Island	Arden Heights	Boro Zone
6	6	Staten Island	Arrochar/Fort Wadsworth	Boro Zone
7	7	Queens	Astoria	Boro Zone

↓ 265 rows | 1.67 seconds runtime

## **Figure 5. Location dataset**

11. Pickup and dropoff location datasets are then combined with combined dataframe (from step 9.). The process of saving and re-loading was also performed multiple times for faster processing.
12. The final dataframe was made by combining all necessary dataframes. The final dataframe contained 630751032 in total.

## **PART 1. Issues in Data Ingestion and Preparation (Solved/Unsolved)**

- The major issue in part 1 was a huge delay from processing of Spark actions after transformation. As a massive amount of data were transformed, every steps of Spark actions took more than an hour. After conducting few experiments, it was found that the process of saving and re-loading dataframes solved this issue (Solved).
- The next issue was that every process of saving and re-loading dataframes also took considerable time but fortunately was able to progress (Solved).
- Also, the cluster was automatically terminated when idling for more than 1 hour (Unsolved).
- Since there were several failures saving final combined dataframe, I decided to first split into two steps of saving final dataframe by separating columns. After repeating saving final dataframe with few columns two times, the final dataframe with all columns was successfully saved (Solved).

## **PART 2. Business Questions**

**The dataframe made from PART1 was converted to a table to execute SQL query.**

1. A. The tables below show year and month from 2015-01 to 2022-12 and total number of trips in each year-month. The total number of trips tends to decrease, as year\_month increases.

	year_month ▲	total_num_trips ▲		year_month ▲	total_num_trips ▲
1	2015-01	12555987			
2	2015-02	12400887	81	2021-09	2456665
3	2015-03	13259877	82	2021-10	2942066
4	2015-04	12953399	83	2021-11	2936122
5	2015-05	13051680	84	2021-12	2708167
6	2015-06	12238777	85	2022-01	2100488
7	2015-07	11436136	86	2022-02	2554960
8	2015-08	11032446	87	2022-03	3071735
9	2015-09	11058388	88	2022-04	3030833
10	2015-10	12200241	89	2022-05	2973224
11	2015-11	11220603	90	2022-06	2954767
12	2015-12	11413782	91	2022-07	2633620
13	2016-01	10832096	92	2022-08	2628150
14	2016-02	11372899	93	2022-09	2619408
15	2016-03	12129922	94	2022-10	3044966
16	2016-04	11866681	95	2022-11	2706794
17	2016-05	11700284	96	2022-12	2809228
18	2016-06	10983327			
19	2016-07	10170942			
20	2016-08	9795886			
21	2016-09	9827608			
22	2016-10	10613182			

**Figure 6. Total number of trips in year-month**

- B. The tables below show year-month from 2015-01 to 2022-12 and the day that has the most trips and their count. Days of week are distributed well but there are slightly less Monday, Tuesday and Sunday.

	year_month ▲	day_of_week ▲	trip_count ▲		year_month ▲	day_of_week ▲	trip_count ▲
1	2015-01	Sat	2395313	79	2021-07	Thu	419139
2	2015-02	Sat	1988795	80	2021-08	Tue	392835
3	2015-03	Sun	2047744	81	2021-09	Thu	433364
4	2015-04	Thu	2234545	82	2021-10	Fri	523492
5	2015-05	Sat	2348827	83	2021-11	Tue	509749
6	2015-06	Tue	1971178	84	2021-12	Thu	472404
7	2015-07	Wed	1948965	85	2022-01	Fri	313526
8	2015-08	Sat	1896798	86	2022-02	Sat	410409
9	2015-09	Wed	1864197	87	2022-03	Thu	531969
10	2015-10	Sat	2229685	88	2022-04	Fri	545700
11	2015-11	Sun	1838437	89	2022-05	Tue	504319
12	2015-12	Thu	1884154	90	2022-06	Thu	530774
13	2016-01	Fri	1982886	91	2022-07	Fri	449134
14	2016-02	Sat	1795028	92	2022-08	Tue	452022
15	2016-03	Thu	2016657	93	2022-09	Thu	496465
16	2016-04	Sat	2222043	94	2022-10	Sat	524026
17	2016-05	Tue	1845006	95	2022-11	Wed	481590
18	2016-06	Thu	1896876	96	2022-12	Thu	492685
19	2016-07	Fri	1777113				
20	2016-08	Wed	1625779				
21	2016-09	Fri	1759543				

**Figure 7. The day of week that had the most trips in each year-month**

C. The tables below show the hour of the day that had the most trips. Apparently, there are the most of trips between 6pm and 7pm.

	year_month ▲	hour_of_day ▲	trip_count ▲		year_month ▲	hour_of_day ▲	trip_count ▲
1	2015-01	07:00 PM	819174	78	2021-06	06:00 PM	177044
2	2015-02	07:00 PM	805602	79	2021-07	06:00 PM	178794
3	2015-03	07:00 PM	858629	80	2021-08	06:00 PM	173182
4	2015-04	07:00 PM	824718	81	2021-09	06:00 PM	183362
5	2015-05	07:00 PM	828931	82	2021-10	06:00 PM	215182
6	2015-06	07:00 PM	776099	83	2021-11	06:00 PM	213497
7	2015-07	07:00 PM	725706	84	2021-12	06:00 PM	190294
8	2015-08	07:00 PM	697121	85	2022-01	06:00 PM	156183
9	2015-09	07:00 PM	710485	86	2022-02	06:00 PM	196942
10	2015-10	07:00 PM	793688	87	2022-03	06:00 PM	234428
11	2015-11	07:00 PM	711955	88	2022-04	06:00 PM	224899
12	2015-12	07:00 PM	724857	89	2022-05	06:00 PM	219181
13	2016-01	06:00 PM	717086	90	2022-06	06:00 PM	216540
14	2016-02	06:00 PM	758943	91	2022-07	06:00 PM	195479
15	2016-03	06:00 PM	795262	92	2022-08	06:00 PM	196025
16	2016-04	07:00 PM	758237	93	2022-09	06:00 PM	186470
17	2016-05	06:00 PM	739218	94	2022-10	06:00 PM	218577
18	2016-06	07:00 PM	690712	95	2022-11	06:00 PM	193335
19	2016-07	06:00 PM	625953	96	2022-12	06:00 PM	191386

**Figure 8. The hour of day that had the most trips in each year-month**



- D. The tables show average of passengers in each year-month. The average of passengers are mostly 1.5, indicating that the number of passengers is 1 or 2 in most of trips.

	year_month ▲	avg_passengers ▲			
1	2015-01	1.5	78	2021-06	1.39
2	2015-02	1.49	79	2021-07	1.42
3	2015-03	1.49	80	2021-08	1.4
4	2015-04	1.49	81	2021-09	1.39
5	2015-05	1.5	82	2021-10	1.39
6	2015-06	1.5	83	2021-11	1.38
7	2015-07	1.5	84	2021-12	1.4
8	2015-08	1.51	85	2022-01	1.35
9	2015-09	1.5	86	2022-02	1.35
10	2015-10	1.49	87	2022-03	1.35
11	2015-11	1.49	88	2022-04	1.37
12	2015-12	1.5	89	2022-05	1.36
13	2016-01	1.49	90	2022-06	1.36
14	2016-02	1.48	91	2022-07	1.39
15	2016-03	1.48	92	2022-08	1.38
16	2016-04	1.49	93	2022-09	1.35
17	2016-05	1.49	94	2022-10	1.35
18	2016-06	1.49	95	2022-11	1.36
19	2016-07	1.5	96	2022-12	1.39

**Figure 9. The Average of Passengers in year-month**

- E. The tables below show the average amount paid per trip in each year-month. The average amount paid tends to increase as year-month increases. For example, in 2015-01, average amount paid is \$13.04, while in 2022-12, average amount paid is \$19.31.

	year_month ▲	avg_amount_per_trip ▲			
1	2015-01	13.04	76	2021-04	15.7
2	2015-02	13.31	77	2021-05	15.78
3	2015-03	13.54	78	2021-06	16.22
4	2015-04	13.72	79	2021-07	16.14
5	2015-05	13.86	80	2021-08	16.13
6	2015-06	13.85	81	2021-09	17.1
7	2015-07	13.64	82	2021-10	16.86
8	2015-08	13.54	83	2021-11	16.9
9	2015-09	13.76	84	2021-12	16.81
10	2015-10	13.89	85	2022-01	15.73
11	2015-11	13.74	86	2022-02	16.33
12	2015-12	13.69	87	2022-03	16.7
13	2016-01	13.19	88	2022-04	16.98
14	2016-02	13.27	89	2022-05	17.34
15	2016-03	13.54	90	2022-06	17.54
16	2016-04	13.77	91	2022-07	16.84
17	2016-05	13.91	92	2022-08	16.93
18	2016-06	14.02	93	2022-09	17.64
19	2016-07	13.8	94	2022-10	17.51
20	2016-08	13.68	95	2022-11	17.4
21	2016-09	14.06	96	2022-12	19.31
22	2016-10	13.86			

**Figure 10. The Average of Amount Paid per Trip in year-month**

- F. The tables show the average amount paid per passenger. Similar to average amount from E, average amount paid per passenger tends to increase, as year-month increases.

	year_month ▲	avg_amount_per_passenger ▲		year_month ▲	avg_amount_per_passenger ▲
1	2015-01	11.04	81	2021-09	14.75
2	2015-02	11.26	82	2021-10	14.48
3	2015-03	11.43	83	2021-11	14.53
4	2015-04	11.59	84	2021-12	14.34
5	2015-05	11.66	85	2022-01	13.77
6	2015-06	11.67	86	2022-02	14.25
7	2015-07	11.47	87	2022-03	14.57
8	2015-08	11.36	88	2022-04	14.66
9	2015-09	11.6	89	2022-05	15.03
10	2015-10	11.71	90	2022-06	15.12
11	2015-11	11.58	91	2022-07	14.42
12	2015-12	11.49	92	2022-08	14.54
13	2016-01	11.14	93	2022-09	15.33
14	2016-02	11.26	94	2022-10	15.18
15	2016-03	11.46	95	2022-11	15.1
16	2016-04	11.62	96	2022-12	16.5

**Figure 11. The Average of Amount Paid per Passenger in year-month**

2. Three tables summarize the average, median, minimum and maximum of trip duration, distance and speed for each yellow and green taxi trip. Green taxi trips have a higher average distance and speed, while yellow taxi trips have a higher average duration. In terms of minimum and maximum values, green and yellow taxi trips have the same rate for duration, distance and speed.

	taxi_colour ▲	avg_trip_duration_minutes ▲	median_trip_duration_minutes ▲	min_trip_duration_minutes ▲	max_trip_duration_minutes ▲
1	green	12.5	10.15	1	50
2	yellow	12.79	10.73	1	50

**Figure 12. Avg, Median, Min, Max Trip Duration for Yellow and Green Taxi**

	taxi_colour ▲	avg_trip_distance_km ▲	median_trip_distance_km ▲	min_trip_distance_km ▲	max_trip_distance_km ▲
1	green	3.97	2.9	0.51	17.99
2	yellow	3.58	2.57	0.51	17.99

**Figure 13. Avg, Median, Min, Max Trip Distance for Yellow and Green Taxi**

	taxi_colour ▲	avg_speed_km_h ▲	median_speed_km_h ▲	min_speed_km_h ▲	max_speed_km_h ▲
1	green	18.99	18	0.67	40.23
2	yellow	17.02	15.89	0.64	40.23

**Figure 14. Avg, Median, Min and Max Trip Speed for Yellow and Green Taxi**

3.

- A. The tables below show the total number of trips based on taxi color, pickup and drop-off borough, month, day and hour. Accordingly, the total number of trips between same boroughs tends to be higher than total number of trips between different boroughs.

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	total_trips ▲
1	green	Bronx	Bronx	01	Fri	01:00 AM	732
2	green	Bronx	Bronx	01	Fri	01:00 PM	1680
3	green	Bronx	Bronx	01	Fri	02:00 AM	553
4	green	Bronx	Bronx	01	Fri	02:00 PM	2076
5	green	Bronx	Bronx	01	Fri	03:00 AM	385
6	green	Bronx	Bronx	01	Fri	03:00 PM	2341
7	green	Bronx	Bronx	01	Fri	04:00 AM	332
8	green	Bronx	Bronx	01	Fri	04:00 PM	2403
9	green	Bronx	Bronx	01	Fri	05:00 AM	322
10	green	Bronx	Bronx	01	Fri	05:00 PM	2512
11	green	Bronx	Bronx	01	Fri	06:00 AM	556
12	green	Bronx	Bronx	01	Fri	06:00 PM	2454
13	green	Bronx	Bronx	01	Fri	07:00 AM	1636
14	green	Bronx	Bronx	01	Fri	07:00 PM	2093
15	green	Bronx	Bronx	01	Fri	08:00 AM	2573
16	green	Bronx	Bronx	01	Fri	08:00 PM	1737
17	green	Bronx	Bronx	01	Fri	09:00 AM	2004
18	green	Bronx	Bronx	01	Fri	09:00 PM	5444

**Figure 15. Total Number of Trips between Same Boroughs**

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	total_trips ▲
4523	green	Bronx	Queens	01	Tue	08:00 AM	18
4524	green	Bronx	Queens	01	Tue	08:00 PM	6
4525	green	Bronx	Queens	01	Tue	09:00 AM	10
4526	green	Bronx	Queens	01	Tue	09:00 PM	7
4527	green	Bronx	Queens	01	Tue	10:00 AM	11
4528	green	Bronx	Queens	01	Tue	10:00 PM	3
4529	green	Bronx	Queens	01	Tue	11:00 AM	8
4530	green	Bronx	Queens	01	Tue	11:00 PM	4
4531	green	Bronx	Queens	01	Tue	12:00 AM	2
4532	green	Bronx	Queens	01	Tue	12:00 PM	9
4533	green	Bronx	Queens	01	Wed	01:00 PM	7
4534	green	Bronx	Queens	01	Wed	02:00 AM	2
4535	green	Bronx	Queens	01	Wed	02:00 PM	12
4536	green	Bronx	Queens	01	Wed	03:00 AM	2
4537	green	Bronx	Queens	01	Wed	03:00 PM	13
4538	green	Bronx	Queens	01	Wed	04:00 AM	3
4539	green	Bronx	Queens	01	Wed	04:00 PM	26
4540	green	Bronx	Queens	01	Wed	05:00 AM	5

**Figure 16. Total Number of Trips between Different Boroughs**

- B. The tables below show average distance of trips based on taxi color, pickup and drop-off borough, month, day and hour. As expected, the average distance between different boroughs tends to be slightly higher than average distance between same boroughs. (Note that average distance between different boroughs can vary depending on pickup and drop-off locations).

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	average_distance ▲
1	green	Bronx	Bronx	01	Fri	01:00 AM	3.253417120218579
2	green	Bronx	Bronx	01	Fri	01:00 PM	3.9987117709523785
3	green	Bronx	Bronx	01	Fri	02:00 AM	3.5664487703435794
4	green	Bronx	Bronx	01	Fri	02:00 PM	3.9422008134874726
5	green	Bronx	Bronx	01	Fri	03:00 AM	3.557184813506494
6	green	Bronx	Bronx	01	Fri	03:00 PM	3.934715994788548
7	green	Bronx	Bronx	01	Fri	04:00 AM	3.793873627710842
8	green	Bronx	Bronx	01	Fri	04:00 PM	3.7503046741573014
9	green	Bronx	Bronx	01	Fri	05:00 AM	3.813436086956522
10	green	Bronx	Bronx	01	Fri	05:00 PM	3.7886503122611446
11	green	Bronx	Bronx	01	Fri	06:00 AM	4.274187058273381
12	green	Bronx	Bronx	01	Fri	06:00 PM	3.571987184841073
13	green	Bronx	Bronx	01	Fri	07:00 AM	3.9551891389975538
14	green	Bronx	Bronx	01	Fri	07:00 PM	3.563592395508836
15	green	Bronx	Bronx	01	Fri	08:00 AM	3.7532836085503254
16	green	Bronx	Bronx	01	Fri	08:00 PM	3.559810048704661

**Figure 17. Average Distance between Same Boroughs**

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	average_distance ▲
5126	green	Bronx	Queens	05	Sun	11:00 AM	12.1263769
5127	green	Bronx	Queens	05	Sun	11:00 PM	10.63326701111111
5128	green	Bronx	Queens	05	Sun	12:00 AM	11.328859522222224
5129	green	Bronx	Queens	05	Sun	12:00 PM	12.136722657142858
5130	green	Bronx	Queens	05	Thu	01:00 AM	15.868092399999998
5131	green	Bronx	Queens	05	Thu	01:00 PM	12.583889271428571
5132	green	Bronx	Queens	05	Thu	02:00 PM	15.038132771428574
5133	green	Bronx	Queens	05	Thu	03:00 AM	16.415267999999998
5134	green	Bronx	Queens	05	Thu	03:00 PM	11.367538104347828
5135	green	Bronx	Queens	05	Thu	04:00 AM	12.774940919999999
5136	green	Bronx	Queens	05	Thu	04:00 PM	12.38752889090909
5137	green	Bronx	Queens	05	Thu	05:00 AM	13.1804946
5138	green	Bronx	Queens	05	Thu	05:00 PM	12.183597877777778
5139	green	Bronx	Queens	05	Thu	06:00 AM	12.514038505882352
5140	green	Bronx	Queens	05	Thu	06:00 PM	12.163592887499998
5141	green	Bronx	Queens	05	Thu	07:00 AM	12.834869676190475

**Figure 18. Average Distance between Different Boroughs**

- C. The results show that the average amount paid between different locations tend to be higher than the average amount between same locations. (Note that the average amount between different locations can vary depending on pickup and drop-off locations).

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	average_amount_per_trip ▲
1	green	Bronx	Bronx	01	Fri	01:00 AM	10.258797814207671
2	green	Bronx	Bronx	01	Fri	01:00 PM	11.737250000000046
3	green	Bronx	Bronx	01	Fri	02:00 AM	10.852405063291162
4	green	Bronx	Bronx	01	Fri	02:00 PM	12.082557803468275
5	green	Bronx	Bronx	01	Fri	03:00 AM	10.492883116883123
6	green	Bronx	Bronx	01	Fri	03:00 PM	12.263267834258935
7	green	Bronx	Bronx	01	Fri	04:00 AM	10.802831325301202
8	green	Bronx	Bronx	01	Fri	04:00 PM	12.744144818976324
9	green	Bronx	Bronx	01	Fri	05:00 AM	9.990683229813659
10	green	Bronx	Bronx	01	Fri	05:00 PM	12.829148089172008
11	green	Bronx	Bronx	01	Fri	06:00 AM	11.32073741007193
12	green	Bronx	Bronx	01	Fri	06:00 PM	12.013251833740858
13	green	Bronx	Bronx	01	Fri	07:00 AM	11.537286063569727
14	green	Bronx	Bronx	01	Fri	07:00 PM	11.751399904443433
15	green	Bronx	Bronx	01	Fri	08:00 AM	11.589766809172215
16	green	Bronx	Bronx	01	Fri	08:00 PM	10.87468048359245

**Figure 19. Average Amount Paid per Trip between Same Boroughs**

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	average_amount_per_trip ▲
8232	green	Brooklyn	Bronx	01	Mon	09:00 PM	47.24
8233	green	Brooklyn	Bronx	01	Mon	10:00 AM	42.56
8234	green	Brooklyn	Bronx	01	Mon	10:00 PM	31.133333333333336
8235	green	Brooklyn	Bronx	01	Mon	12:00 AM	34.8
8236	green	Brooklyn	Bronx	01	Sat	01:00 AM	45.96
8237	green	Brooklyn	Bronx	01	Sat	02:00 AM	33.95
8238	green	Brooklyn	Bronx	01	Sat	02:00 PM	52.01
8239	green	Brooklyn	Bronx	01	Sat	03:00 AM	30.8
8240	green	Brooklyn	Bronx	01	Sat	04:00 AM	33.55
8241	green	Brooklyn	Bronx	01	Sat	05:00 AM	33.308
8242	green	Brooklyn	Bronx	01	Sat	05:00 PM	31.799999999999997
8243	green	Brooklyn	Bronx	01	Sat	06:00 PM	35.18
8244	green	Brooklyn	Bronx	01	Sat	07:00 PM	36.95
8245	green	Brooklyn	Bronx	01	Sat	08:00 AM	52.58
8246	green	Brooklyn	Bronx	01	Sat	08:00 PM	40.13
8247	green	Brooklyn	Bronx	01	Sat	09:00 PM	36.8
8248	green	Brooklyn	Bronx	01	Sat	10:00 AM	42.36

**Figure 20. Average Amount Paid per Trip between Different Boroughs**

- D. Total amounts paid between same boroughs are higher than total amounts paid between different boroughs. Because there are larger number of trips between same boroughs than different boroughs.

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	total_amount_paid ▲
1	green	Bronx	Bronx	01	Fri	01:00 AM	7509.440000000015
2	green	Bronx	Bronx	01	Fri	01:00 PM	19718.580000000078
3	green	Bronx	Bronx	01	Fri	02:00 AM	6001.380000000013
4	green	Bronx	Bronx	01	Fri	02:00 PM	25083.390000000138
5	green	Bronx	Bronx	01	Fri	03:00 AM	4039.760000000002
6	green	Bronx	Bronx	01	Fri	03:00 PM	28708.310000000165
7	green	Bronx	Bronx	01	Fri	04:00 AM	3586.539999999999
8	green	Bronx	Bronx	01	Fri	04:00 PM	30624.180000000106
9	green	Bronx	Bronx	01	Fri	05:00 AM	3216.999999999998
10	green	Bronx	Bronx	01	Fri	05:00 PM	32226.820000000087
11	green	Bronx	Bronx	01	Fri	06:00 AM	6294.329999999993
12	green	Bronx	Bronx	01	Fri	06:00 PM	29480.520000000066
13	green	Bronx	Bronx	01	Fri	07:00 AM	18875.000000000073
14	green	Bronx	Bronx	01	Fri	07:00 PM	24595.680000000106
15	green	Bronx	Bronx	01	Fri	08:00 AM	29820.470000000107
16	green	Bronx	Bronx	01	Fri	08:00 PM	18889.320000000083

**Figure 21. Total Amount Paid between Same Boroughs based on taxi color, month, day and hour**

	taxi_colour ▲	Pick_Borough ▲	Drop_Borough ▲	pickup_month ▲	day_of_week ▲	hour_of_day ▲	total_amount_paid ▲
5088	green	Bronx	Queens	05	Sat	03:00 PM	901.75
5089	green	Bronx	Queens	05	Sat	04:00 PM	807.8300000000002
5090	green	Bronx	Queens	05	Sat	05:00 AM	127.38000000000001
5091	green	Bronx	Queens	05	Sat	05:00 PM	816.9
5092	green	Bronx	Queens	05	Sat	06:00 AM	152.8
5093	green	Bronx	Queens	05	Sat	06:00 PM	788.26
5094	green	Bronx	Queens	05	Sat	07:00 AM	288
5095	green	Bronx	Queens	05	Sat	07:00 PM	657.61
5096	green	Bronx	Queens	05	Sat	08:00 AM	385.77000000000004
5097	green	Bronx	Queens	05	Sat	08:00 PM	619.4300000000001
5098	green	Bronx	Queens	05	Sat	09:00 AM	426.26
5099	green	Bronx	Queens	05	Sat	09:00 PM	440.79000000000001
5100	green	Bronx	Queens	05	Sat	10:00 AM	265.38
5101	green	Bronx	Queens	05	Sat	10:00 PM	725.46
5102	green	Bronx	Queens	05	Sat	11:00 AM	626.59
5103	green	Bronx	Queens	05	Sat	11:00 PM	607.4000000000001

**Figure 22. Total Amount Paid between Different Boroughs based on taxi color, month, day and hour**

4. 63.9% of trips, where drivers received tips.

percentage_of_trips_with_tips ▲	
1	63.9

**Figure 23. Percentage of Trips Where Drivers Received Tips**

5. 6.9% of trips, where drivers received at least \$5 tips.

percentage_with_tips_over_5 ▲	
1	6.9

**Figure 24. Percentage of Trips Where Drivers Received at least \$5 Tips**

6. 30-60 mins bin has the highest average distance per dollar, while under 5 mins bin has the lowest average distance per dollar. Also, under 5 mins bin has the highest average speed.

	trip_duration_bins ▲	avg_distance_per_dollar ▲		trip_duration_bins ▲	avg_speed_km_h ▲
1	10-20 Mins	0.25	1	10-20 Mins	16.42
2	20-30 Mins	0.29	2	20-30 Mins	17.71
3	30-60 Mins	0.31	3	30-60 Mins	17.16
4	5-10 Mins	0.21	4	5-10 Mins	16.77
5	Under 5 Mins	0.17	5	Under 5 Mins	19.78

**Figure 25. Average of distance per dollar and average speed in trip duration category**

7. I would advise a taxi driver to target 30-60 bin, to maximize the income because it has the highest average distance per dollar, indicating that passengers are paying more for longer trips. Consequently, it is likely that longer trips lead to higher fares and higher incomes for drivers.

### **PART 3. Machine Learning**

1.
  - A. A baseline model was built using the average of total amount as a prediction. The target variable is "total\_amount" of a trip. The average of total amount obtained was 14.307 and it was added as a new column into the final dataframe. Subsequently, "RegressionEvaluator" function was used to calculate its RMSE score.



```
# Using RegressionEvaluator to obtain RMSE score.
from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator(labelCol='total_amount', predictionCol='baseline_prediction', metricName='rmse')
rmse = evaluator.evaluate(baseline_prediction)
```

▸ (1) Spark Jobs

💡 1

Command took 10.61 minutes -- by dlatldus12@gmail.com at 04/10/2023, 8:45:16 pm on My Cluster

---

md 5

```
# As expected, the baseline RMSE is extremely high, resulting in a poor performance.
print("Baseline RMSE:", rmse)
```

Baseline RMSE: 196.29815476684607

**Figure 26. Baseline Model RMSE Score**

The baseline model RMSE obtained was 196.298, which indicates that it is not a valid model for predicting total amount of trips.

- B. Features that were used as predictor variables include “trip\_distance\_km”, “trip\_duration\_sec”, “speed\_km\_h”, “passenger\_count”, “tip\_amount”, “tolls\_amount”, “Pick\_Borough” and “Drop\_Borough”. “Pick\_Borough” and “Drop\_Borough” were categorical features and therefore converted to numeric values, using “OneHotEncoder”.

```
from pyspark.ml import Pipeline
from pyspark.sql.functions import col

string_indexer = [StringIndexer(inputCol=col, outputCol=col + "_index") for col in cat_cols]
encoder = [OneHotEncoder(inputCol=col + "_index", outputCol=col + "_encoded") for col in cat_cols]

pipeline = Pipeline(stages=string_indexer + encoder)
pipeline_model = pipeline.fit(df_taxi)
df_taxi = pipeline_model.transform(df_taxi)
```

▸ (4) Spark Jobs

▸ 📄 df\_taxi: pyspark.sql.dataframe.DataFrame = [taxi\_colour: string, pickup\_datetime: timestamp ... 14 more fields]

Command took 7.37 minutes -- by dlatldus12@gmail.com at 04/10/2023, 3:05:39 pm on My Cluster

---

Cmd 12

```
# Finalising features, that will be used as predictor values.
features = ["trip_distance_km", "trip_duration_sec", "speed_km_h", "passenger_count", "tip_amount", "tolls_amount", "Pick_Borough_index", "Drop_Borough_index"]
```

Command took 0.12 seconds -- by dlatldus12@gmail.com at 04/10/2023, 3:11:13 pm on My Cluster

---

Cmd 13

```
# Used VectorAssembler method to assemble predictor variables and making into a new column.
assembler = VectorAssembler(inputCols=features, outputCol="features")
df = assembler.transform(df_taxi)
```

**Figure 27. Feature Selection, Conversion and Assemble**

- C. Two regression model algorithms I chose were “LinearRegression” and “RandomForestRegressor”.

**Linear Regression:** The rationale behind choosing linear regression algorithm is that it is simple regression algorithm and is therefore generally fast to train (i.e., taking less processing time) and also suitable for a large dataset. Furthermore, some features that are



used as predictor variables are linearly correlated (e.g., distance and speed and duration and speed) and there are correlations between target variable (“total\_amount”) and features (e.g., “tip\_amount” and “tolls\_amount”). Thus, linear regression algorithm was chosen as one of two regression algorithms.

**Random Forest Regressor:** Random Forest Regressor is an optimal regression algorithm for big data, as it can scale and handle a large number of data and features. Also, given that it is a non-parametric regression algorithm, it is likely to perform well on a wide range of data and a number of features. Furthermore, its flexibility that allows to set the number of trees for a large dataset increase the performance of models and allows to try different models with different hyperparameters.

## D. Results

**Linear Regression Model:** Default Linear Regression algorithm was trained on training dataset and testing dataset (“total\_ amount” of October/November/December 2022 Trips) was predicted. RMSE score obtained was 2.826, which indicates much greater performance, compared to baseline model (196.298).

```
+-----+-----+
|total_amount|      prediction|
+-----+-----+
|          13.0|11.634977789639944|
|           9.8| 9.564366515002567|
|          11.3|11.310397794656271|
|          19.15| 19.60949798823612|
|           16.8| 16.27264649414458|
|           25.8| 23.20902496237086|
|          10.56| 9.493906262048323|
|          37.56| 36.18129064189908|
|          29.16|28.490802681370624|
|          21.96|21.798720286335374|
+-----+-----+
only showing top 10 rows
```

Command took 3.18 seconds -- by dlatldus12@gmail.com at 04/10/2023, 9:23:03 pm on My Cluster

Cmd 31

```
evaluator_lr = RegressionEvaluator(labelCol='total_amount', predictionCol='prediction', metricName='rmse')
rmse_lr = evaluator_lr.evaluate(test_preds_lr)
print("Linear Regression RMSE:", rmse_lr)
```

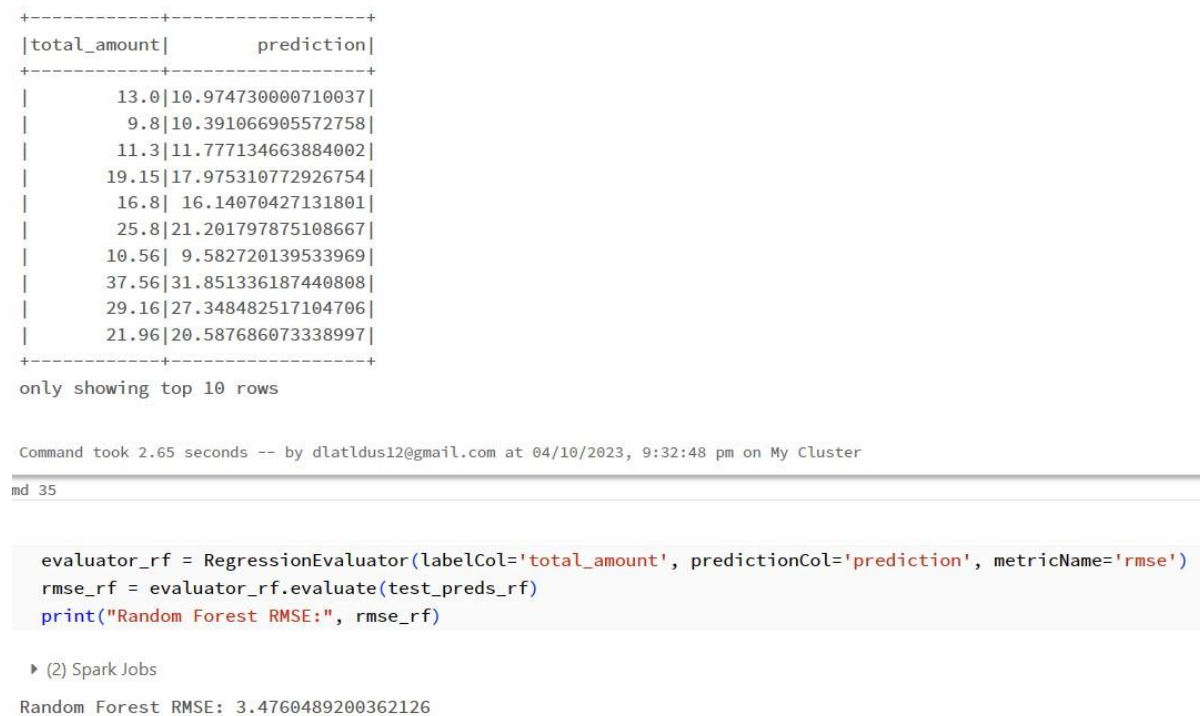
▶ (2) Spark Jobs

Linear Regression RMSE: 2.825784876817214

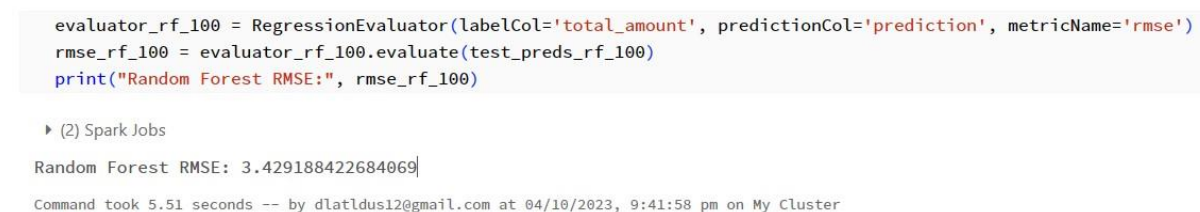
**Figure 28. Results of Linear Regression Model**

**Random Forest Regressor Models:** Two different Random Forest models were trained and assessed, one with default model and the other with 100 number of trees. RMSE score obtained for default model was 3.476, while RMSE score for 100 tress was 3.429. While

Random Forest Regressor models showed relatively poorer performance than linear regression model, they still showed a great performance and beat a baseline model.



**Figure 29. Results of Random Forest Regressor Default Model**



**Figure 30. Results of Random Forest Regressor Model with 100 Trees**

## PART 3. Machine Learning Issues (Solved/Unsolved)

- Faced difficulties when training models, as it took more than 1 hour each for training a huge amount of data. I decided to reduce the amount of training and testing data (80000, 20000) for faster processing (Solved).

## 4 REFERENCE

---

TLC Trip Records User Guide. (2019). Available at:

[https://www.nyc.gov/assets/tlc/downloads/pdf/trip\\_record\\_user\\_guide.pdf](https://www.nyc.gov/assets/tlc/downloads/pdf/trip_record_user_guide.pdf).

www.nyc.gov. (n.d.). *TLC Trip Record Data - TLC*. [online] Available at:

<https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>.