# NYC Taxi Bigdata processing & Machine learning modelling with Databricks

Big Data Engineering | Autumn, 2024

Taekjin Jeong (25099654)

<06.10.2024>

# Table of Contents

## 1. Executive Summary

This project aims to process big data with PARQUET based on NYC taxi trip data. After processing data, it presents the business analyses and predictive modelling results, focusing on the period between 2015 and 2021. The objectives derive insights from the data and build machine learning models that could accurately predict a single trip total amount. By leveraging business analysis and machine learning techniques, the project focuses on developing a robust solution that not only forecasts taxi fares but can also generalize well to unseen data.

## 2. Introduction

This project processes NYC taxi data stored in PARQUET format using Apache Spark in Databricks. In Part 1, we will mount Azure storage to Spark, retrieve the PARQUET files, and prepare the data by merging them into a single PARQUET file after data processing. Part 2 focuses on leveraging SparkSQL to address business-related queries. Finally, Part 3 involves developing a machine learning model to predict the total amount for each ride, with Root Mean Squared Error (RMSE) serving as the evaluation metric for model performances.

## 3. Data Ingestion (Part 1)

*Corresponding file: Part_1.ipynb, Part_1.html*

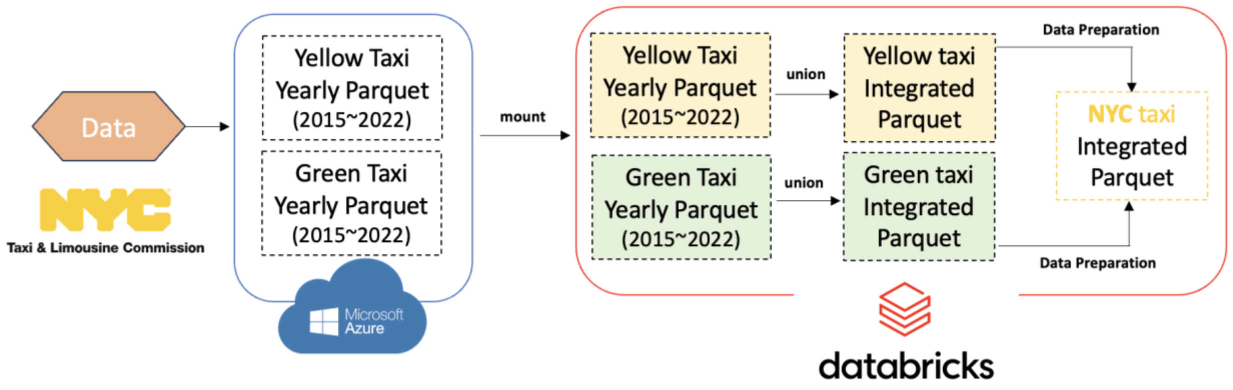

Data
NYC
Taxi & Limousine Commission
Yellow Taxi Yearly Parquet (2015~2022)
Green Taxi Yearly Parquet (2015~2022)
Microsoft Azure
mount
Yellow Taxi Yearly Parquet (2015~2022)
union
Yellow taxi Integrated Parquet
Green Taxi Yearly Parquet (2015~2022)
union
Green taxi Integrated Parquet
Data Preparation
NYC taxi Integrated Parquet
Data Preparation
databricks

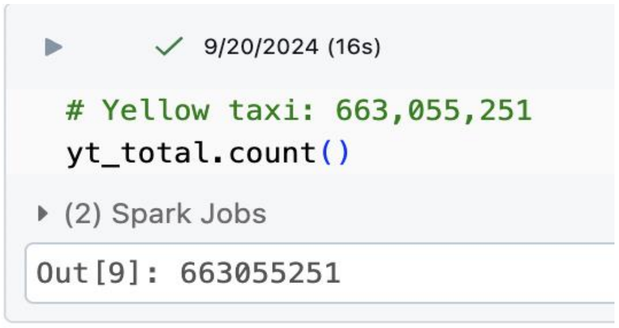
*Figure 1. Data ingestion process*

The NYC Taxi and Limousine Commission (TLC) collects yellow and green taxi trip records with 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 dataset is provided in a parquet format. Figure 1 shows the data ingestion process. Firstly, we loaded yellow and green taxi parquet files from 2015 to 2022 into Azure Blob Storage and made a copy of it into Databricks File System (DBFS). dbutils.fs.mount function was used to attach the Azure storage. When migrating the dataset into DBFS, we did not write the exact path; hence files were saved in the Azure storage. The destination path needs to be set /dbfs/ at the beginning of the path to save the PARQUET file in DBFS.

**Example (The destination Path)**

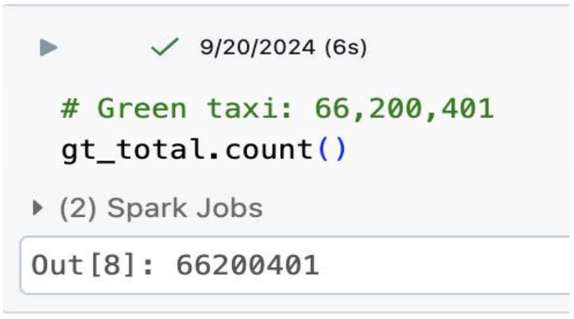
```
"/dbfs/mnt/bde-assignment2/yearly_data/{file_path.split('/')[-1]}"
```

The PARQUET files that have been moved to DBFS are converted into DataFrames, and the union function is used to merge them into a single PARQUET file. This single PARQUET file is efficient for storage management and further processing. Figure 2 indicates the total number of rows for each taxi colour (yellow and green). The yellow taxi parquet includes 663,055,251 rows, and the green taxi has 66,200,401 rows. It indicates that the number of yellow taxi rides is approximately ten times greater than that of green taxi rides during the same period.



```
9/20/2024 (16s)
# Yellow taxi: 663,055,251
yt_total.count()
(2) Spark Jobs
Out[9]: 663055251
```

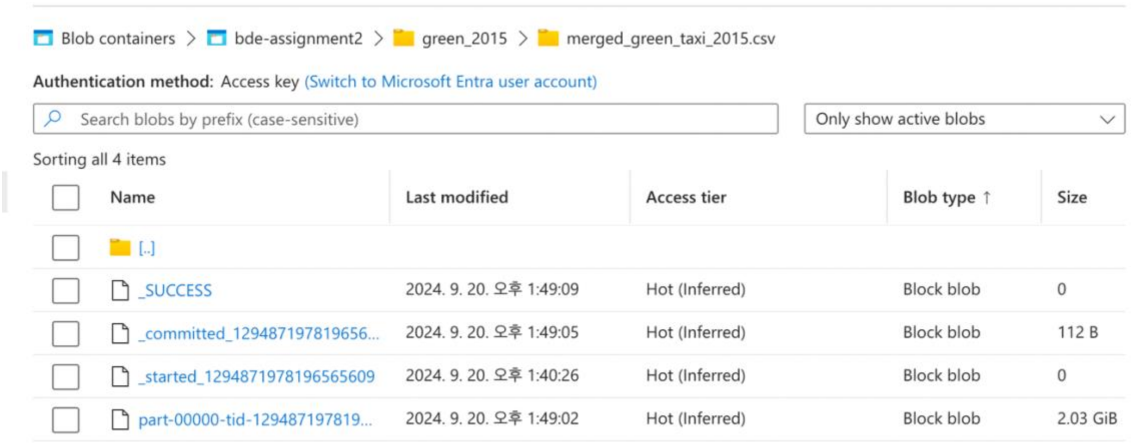
*Figure 2(a). Yellow taxi parquet*



```
9/20/2024 (6s)
# Green taxi: 66,200,401
gt_total.count()
(2) Spark Jobs
Out[8]: 66200401
```

*Figure 2(b). Green taxi parquet*

To verify the efficiency of Parquet files, the 2015 green taxi Parquet file was converted to CSV format and stored in Azure. Figure 3 shows the result of storing the file in Azure: the original Parquet file, which was 385 MB, expanded to 2.03 GB when converted to CSV format. This highlights one of the critical advantages of Parquet files: their ability to compress and store data more efficiently compared to CSV files. Parquet uses columnar storage, which reduces file size and optimises performance, mainly when working with large datasets in distributed systems such as Spark.


Blob containers > bde-assignment2 > green_2015 > merged_green_taxi_2015.csv

Authentication method: Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive) | Only show active blobs

Sorting all 4 items

| Name | Last modified | Access tier | Blob type ↑ | Size |
|---|---|---|---|---|
| [..] | | | | |
| _SUCCESS | 2024. 9. 20. 오후 1:49:09 | Hot (Inferred) | Block blob | 0 |
| _committed_129487197819656... | 2024. 9. 20. 오후 1:49:05 | Hot (Inferred) | Block blob | 112 B |
| _started_1294871978196565609 | 2024. 9. 20. 오후 1:40:26 | Hot (Inferred) | Block blob | 0 |
| part-00000-tid-129487197819... | 2024. 9. 20. 오후 1:49:02 | Hot (Inferred) | Block blob | 2.03 GiB |

*Figure 3. The green taxi csv file in Azure*

## 4. Data preparation (Part 1)

*Corresponding file: Part_1.ipynb, Part_1.html*

Before merging green and yellow taxi parquet with a location file, this project performed Data preparation. Table 1 contains the tasks performed for data preparation. During the data preparation process, we identified a significant amount of erroneous information in the dataset.

*Table 1. Data prepartion*

| Data preparation | Description | Outcome |
|---|---|---|
| Data cleansing | Trips finishing before the starting time | Filtered out trips with invalid times. |
| | Trip_distance is less than 0 | Filtered out trips with invalid distance. |
| | Fare_amount and Total_amount are less than 0 | Filtered out trips with invalid Fare amount and total amount |
| | Speed is negative or over 200 | Filtered out trips with invalid Speed |
| | Duration is under 1 min or over 120 minutes, and distance is under | Filtered out outlier trips |

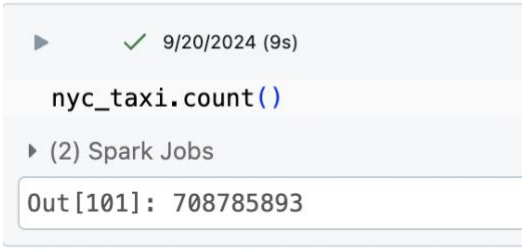
| Data preparation | Description | Outcome |
|---|---|---|
| | 0.5km or over 80km (Green taxi applied 100km) | |
| Feature engineering | Added trip duration column | Calculate pick-up datetime and drop-off datetime |
| | Added speed_kmh column | Change the speed unit from MPH to KMH |
| | Replace passanger_count 0 with mode 1 | Apply the mode value of 1 to the null values in the passenger count. |
| | Added Taxi_type column | "Yellow" and "Green" type is added to identify different taxi type. |
| Feature engineering | Replace the Pickup and Dropoff columns in the Green table and Yellow taxi with pickup_datetime and dropoff_datetime | |
| | Add columns that are not in both datasets and fill them with null values | Make same schema to combine two taxi datasets |

## Example code (Data preparation)

```
# Remove trips finishing before the starting time
gt_df = gt_df.filter(col("lpep_dropoff_datetime") > col("lpep_pickup_datetime"))

# Remove Trip_distance is less than 0
gt_df = gt_df.filter(col("Trip_distance") > 0)

# Remove Fare_amount is less than 0
gt_df = gt_df.filter(col("Fare_amount") > 0)

# Remove Total_amount is less than 0
gt_df = gt_df.filter(col("Total_amount") > 0)
```

Notably, we found that the most common number of taxi passengers is 1, the average taxi fare is approximately $13, and the average speed is about 25 km while covering 4.63 km. After completing the two-phase taxi combination, we performed a join with the location data to create a new DataFrame. To differentiate between pickup and dropoff locations, we conducted two join operations, using "borough" for "pickup_borough" and "borough" for "dropoff_borough." Once all data preparation processes were finalised, the resulting dataset contained 70,878,593 rows (Figure 4).

The resulting DataFrame was then saved as nyc_taxi_final.parquet to DBFS.



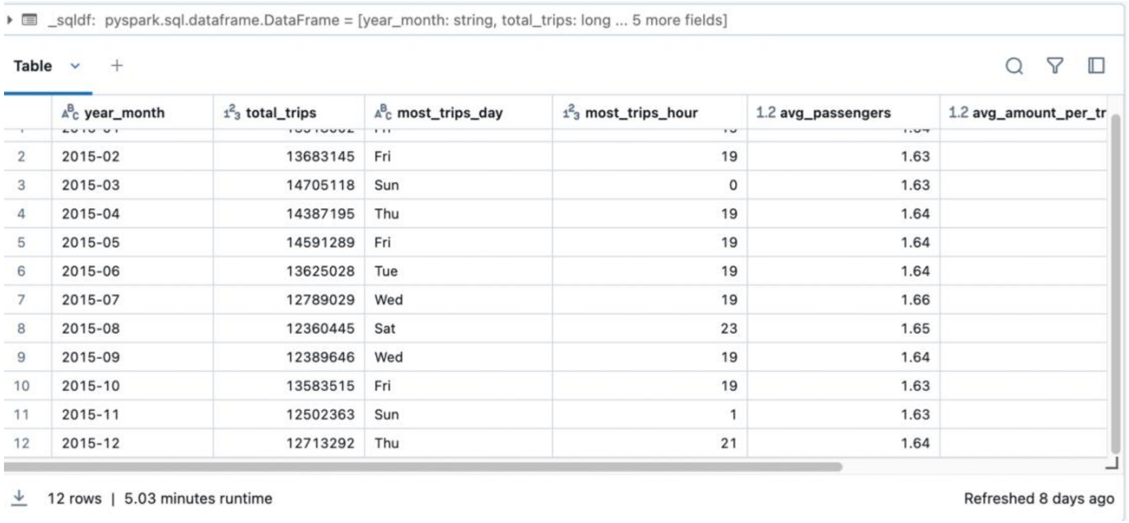
```
9/20/2024 (9s)
nyc_taxi.count()
(2) Spark Jobs
Out[101]: 708785893
```

*Figure 4. The final dataset result*

## 5. Business Analyses (Part 2)

*Corresponding file: Part_2.ipynb, Part_2.html*

- Q1) For each year and month, get results of total number trips, the busiest day and hours, the average of passengers, amount paid per trip and amount paid per passenger.



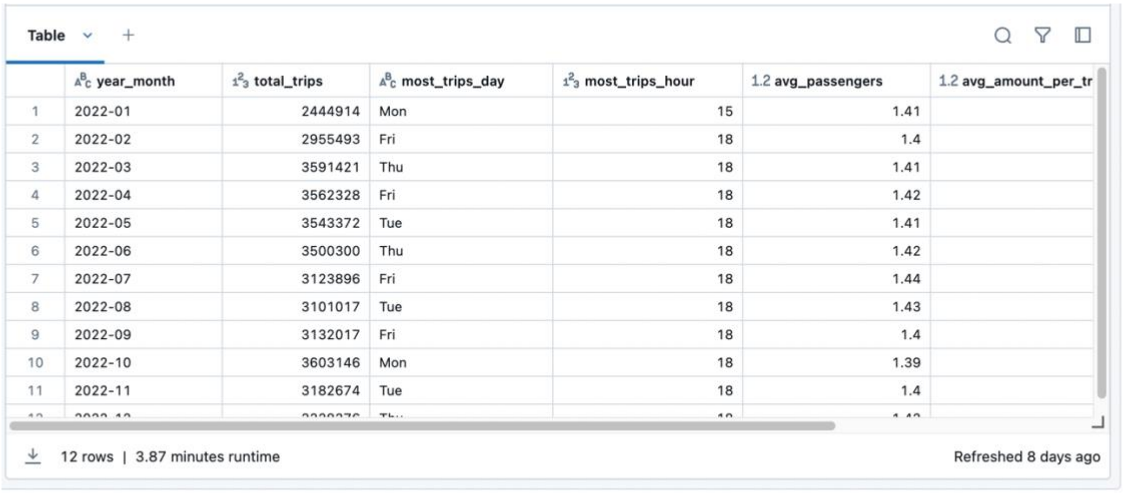
_sqldf: pyspark.sql.dataframe.DataFrame = [year_month: string, total_trips: long ... 5 more fields]

| | year_month | total_trips | most_trips_day | most_trips_hour | avg_passengers | avg_amount_per_tr |
|---|---|---|---|---|---|---|
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | |
| 2 | 2015-02 | 13683145 | Fri | 19 | 1.63 | |
| 3 | 2015-03 | 14705118 | Sun | 0 | 1.63 | |
| 4 | 2015-04 | 14387195 | Thu | 19 | 1.64 | |
| 5 | 2015-05 | 14591289 | Fri | 19 | 1.64 | |
| 6 | 2015-06 | 13625028 | Tue | 19 | 1.64 | |
| 7 | 2015-07 | 12789029 | Wed | 19 | 1.66 | |
| 8 | 2015-08 | 12360445 | Sat | 23 | 1.65 | |
| 9 | 2015-09 | 12389646 | Wed | 19 | 1.64 | |
| 10 | 2015-10 | 13583515 | Fri | 19 | 1.63 | |
| 11 | 2015-11 | 12502363 | Sun | 1 | 1.63 | |
| 12 | 2015-12 | 12713292 | Thu | 21 | 1.64 | |

12 rows | 5.03 minutes runtime

Refreshed 8 days ago

*Figure 5. Question 1 result (2015)*



Table

| | year_month | total_trips | most_trips_day | most_trips_hour | avg_passengers | avg_amount_per_tr |
|---|---|---|---|---|---|---|
| 1 | 2022-01 | 2444914 | Mon | 15 | 1.41 | |
| 2 | 2022-02 | 2955493 | Fri | 18 | 1.4 | |
| 3 | 2022-03 | 3591421 | Thu | 18 | 1.41 | |
| 4 | 2022-04 | 3562328 | Fri | 18 | 1.42 | |
| 5 | 2022-05 | 3543372 | Tue | 18 | 1.41 | |
| 6 | 2022-06 | 3500300 | Thu | 18 | 1.42 | |
| 7 | 2022-07 | 3123896 | Fri | 18 | 1.44 | |
| 8 | 2022-08 | 3101017 | Tue | 18 | 1.43 | |
| 9 | 2022-09 | 3132017 | Fri | 18 | 1.4 | |
| 10 | 2022-10 | 3603146 | Mon | 18 | 1.39 | |
| 11 | 2022-11 | 3182674 | Tue | 18 | 1.4 | |

12 rows | 3.87 minutes runtime

Refreshed 8 days ago

*Figure 6.. Question 1 result (2022)*

- To maximise query performance, we executed queries for a single year to answer the question. Figure 5, 6 shows the sample output of query results (2015, 2022). Figure 7 indicates that Friday is the busiest day for taxi drivers. Moreover, it was analysed that the busiest hours for taxi drivers are not during the morning rush hour but rather around 6 p.m. and 7 p.m (Figure 8). Furthermore, the average number of passengers is 1.53, the amount paid per trip is $18.59, the amount paid per passenger is $15.18 and the average number of trips in a month is 7,383,162.64. Through these results, it is possible to identify the days and times when taxi passengers most frequently use the service, as well as the average amount they spend.

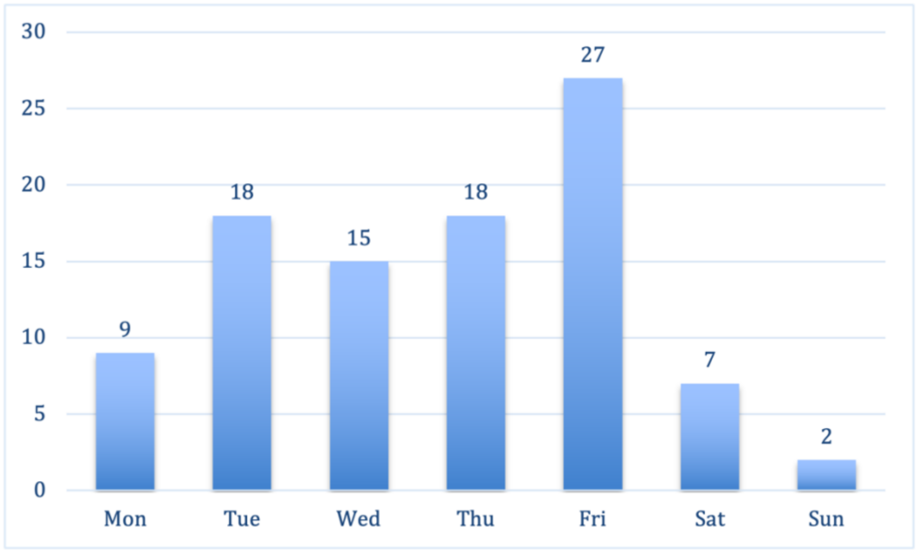


*Figure 7. Comparison of taxi trips by a day of week*

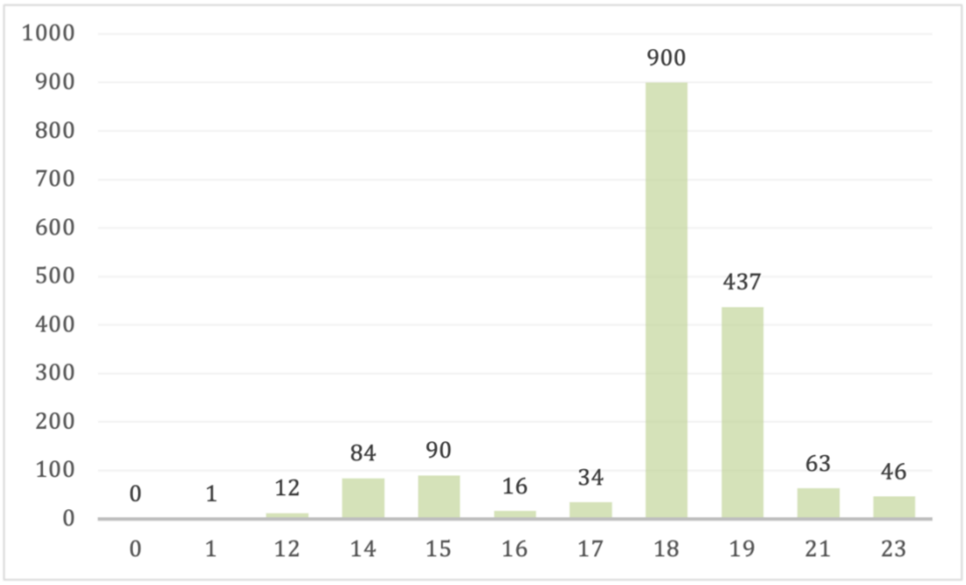
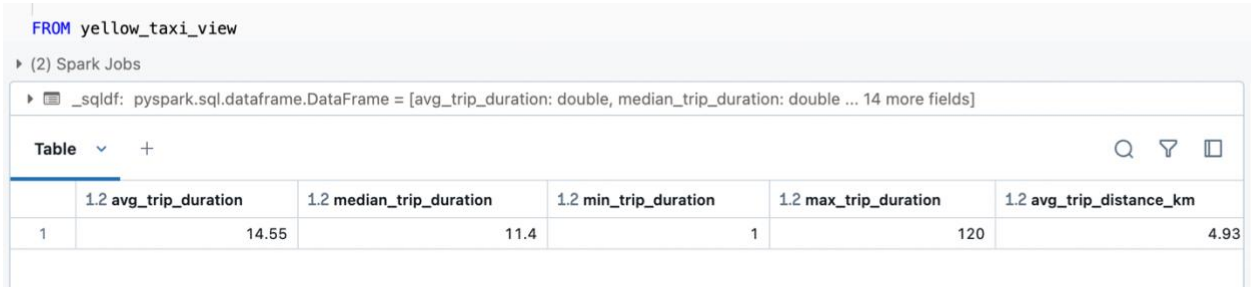


*Figure 8. Comparison of taxi trips by hour*

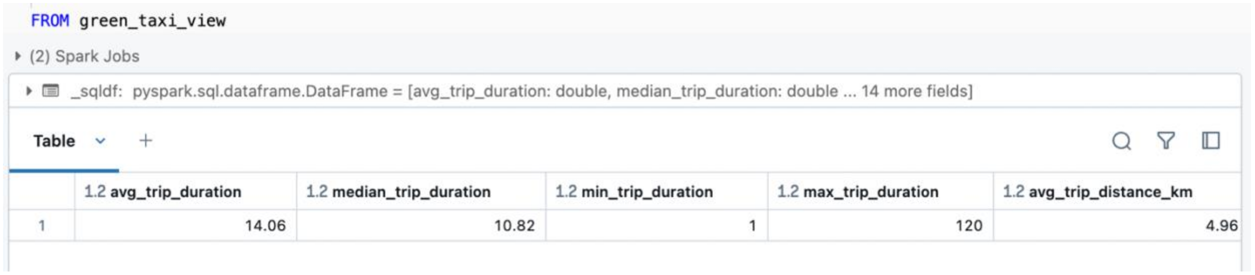
- Q2) Provide information of the average, median, minimum, and maximum duration, distances and speed for yellow and green taxi.



```
FROM yellow_taxi_view
▸ (2) Spark Jobs
▸ _sqldf: pyspark.sql.dataframe.DataFrame = [avg_trip_duration: double, median_trip_duration: double ... 14 more fields]
```

| | 1.2 avg_trip_duration | 1.2 median_trip_duration | 1.2 min_trip_duration | 1.2 max_trip_duration | 1.2 avg_trip_distance_km |
|---|---|---|---|---|---|
| 1 | 14.55 | 11.4 | 1 | 120 | 4.93 |

*Figure 9. Question 2 result (yellow taxi)*



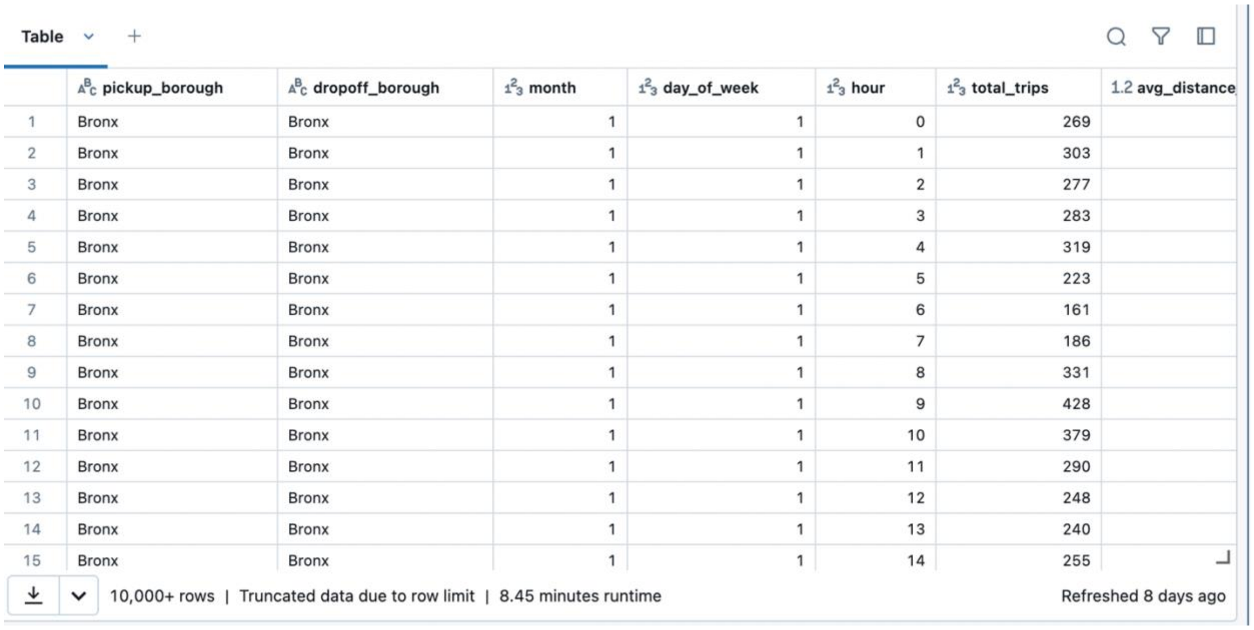
```
FROM green_taxi_view
▸ (2) Spark Jobs
▸ _sqldf: pyspark.sql.dataframe.DataFrame = [avg_trip_duration: double, median_trip_duration: double ... 14 more fields]
```

| | 1.2 avg_trip_duration | 1.2 median_trip_duration | 1.2 min_trip_duration | 1.2 max_trip_duration | 1.2 avg_trip_distance_km |
|---|---|---|---|---|---|
| 1 | 14.06 | 10.82 | 1 | 120 | 4.96 |

*Figure 10. Question 2 result (green taxi)*

- Figure 9, 10 shows the results (Yellow and Green taxi). It indicates that the yellow taxi's average of trip duration is slightly higher than the green taxi. Minimum and max duration was same because of data preparation. It was applied to trip distances. The average of speed in green taxi was higher than yellow taxi. The minimum speed is below 1, which is unrealistic in real-world scenarios. In the future experiment, feature engineering will need to adjust these outlier speeds to align with real-world conditions.

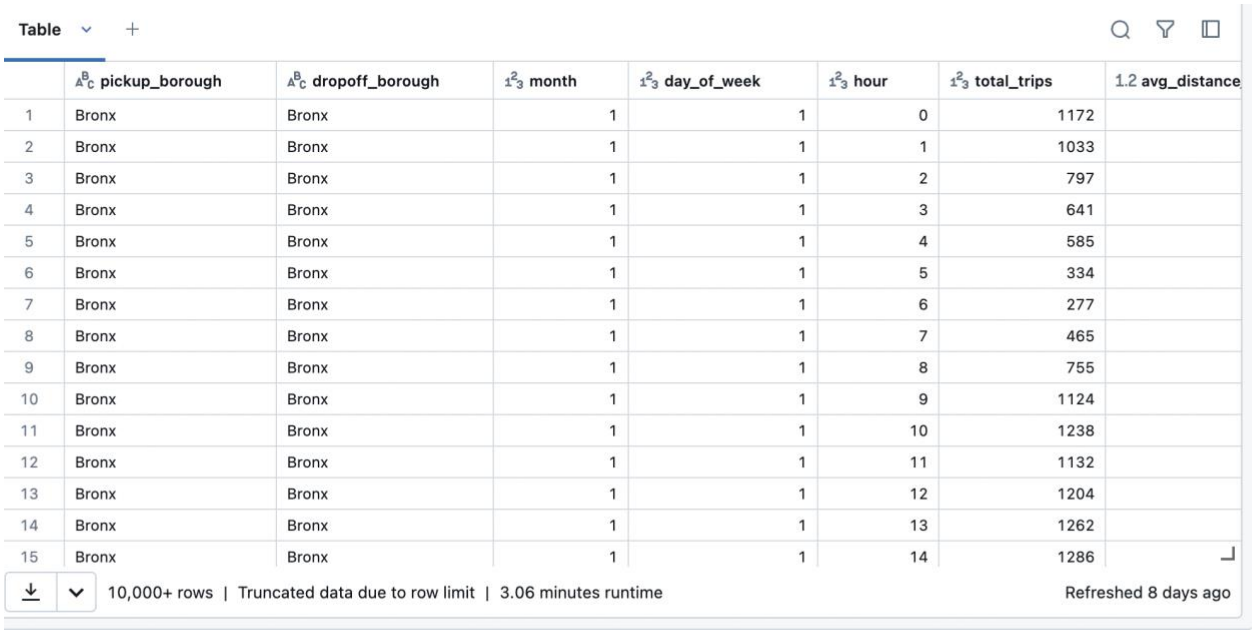
- Q3) Provide total number of trips, average distances, average amount per trip, and total amount for each pair of pick up and drop off locations for yellow and green taxi.



| | pickup_borough | dropoff_borough | month | day_of_week | hour | total_trips | avg_distance |
|---|---|---|---|---|---|---|---|
| 1 | Bronx | Bronx | 1 | 1 | 0 | 269 | |
| 2 | Bronx | Bronx | 1 | 1 | 1 | 303 | |
| 3 | Bronx | Bronx | 1 | 1 | 2 | 277 | |
| 4 | Bronx | Bronx | 1 | 1 | 3 | 283 | |
| 5 | Bronx | Bronx | 1 | 1 | 4 | 319 | |
| 6 | Bronx | Bronx | 1 | 1 | 5 | 223 | |
| 7 | Bronx | Bronx | 1 | 1 | 6 | 161 | |
| 8 | Bronx | Bronx | 1 | 1 | 7 | 186 | |
| 9 | Bronx | Bronx | 1 | 1 | 8 | 331 | |
| 10 | Bronx | Bronx | 1 | 1 | 9 | 428 | |
| 11 | Bronx | Bronx | 1 | 1 | 10 | 379 | |
| 12 | Bronx | Bronx | 1 | 1 | 11 | 290 | |
| 13 | Bronx | Bronx | 1 | 1 | 12 | 248 | |
| 14 | Bronx | Bronx | 1 | 1 | 13 | 240 | |
| 15 | Bronx | Bronx | 1 | 1 | 14 | 255 | |

10,000+ rows | Truncated data due to row limit | 8.45 minutes runtime Refreshed 8 days ago

*Figure 11. Question 3 result (yellow taxi)*



| | pickup_borough | dropoff_borough | month | day_of_week | hour | total_trips | avg_distance |
|---|---|---|---|---|---|---|---|
| 1 | Bronx | Bronx | 1 | 1 | 0 | 1172 | |
| 2 | Bronx | Bronx | 1 | 1 | 1 | 1033 | |
| 3 | Bronx | Bronx | 1 | 1 | 2 | 797 | |
| 4 | Bronx | Bronx | 1 | 1 | 3 | 641 | |
| 5 | Bronx | Bronx | 1 | 1 | 4 | 585 | |
| 6 | Bronx | Bronx | 1 | 1 | 5 | 334 | |
| 7 | Bronx | Bronx | 1 | 1 | 6 | 277 | |
| 8 | Bronx | Bronx | 1 | 1 | 7 | 465 | |
| 9 | Bronx | Bronx | 1 | 1 | 8 | 755 | |
| 10 | Bronx | Bronx | 1 | 1 | 9 | 1124 | |
| 11 | Bronx | Bronx | 1 | 1 | 10 | 1238 | |
| 12 | Bronx | Bronx | 1 | 1 | 11 | 1132 | |
| 13 | Bronx | Bronx | 1 | 1 | 12 | 1204 | |
| 14 | Bronx | Bronx | 1 | 1 | 13 | 1262 | |
| 15 | Bronx | Bronx | 1 | 1 | 14 | 1286 | |

10,000+ rows | Truncated data due to row limit | 3.06 minutes runtime Refreshed 8 days ago

*Figure 12. Question 3 result (green taxi)*

- Figure 11, 12 shows the sample output of query results (Yellow and Green taxi). When the pickup_borough and dropoff_borough are the same, the number of taxi trips increases. Additionally, the frequency of taxi rides varies depending on the day of the week and the time of day. Figure 13 is a line graph where both the pickup_borough and dropoff_borough are Bronx in January. It indicates that taxi usage is highest during the morning rush hour at 8 AM, then gradually decreases as the day progresses, before increasing again starting from 1 PM.

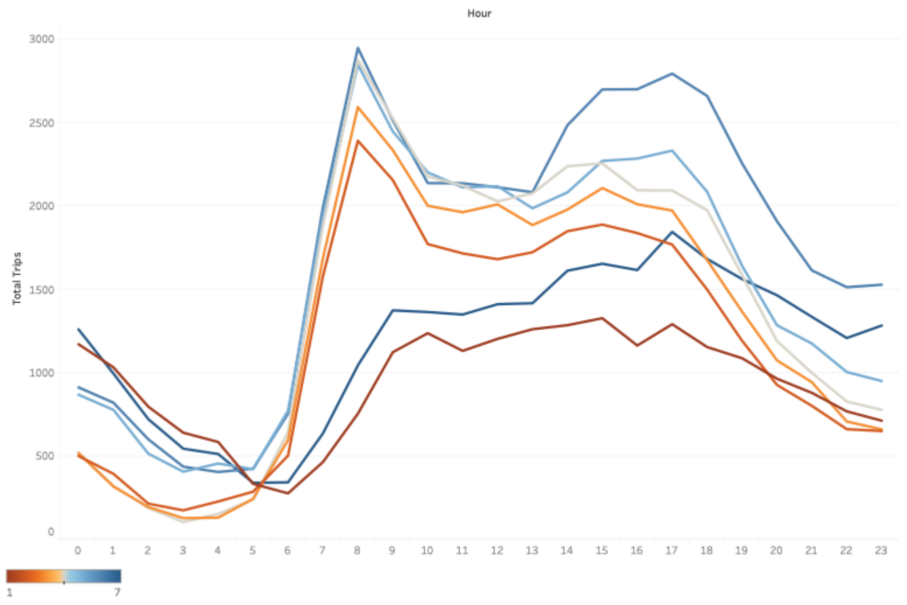
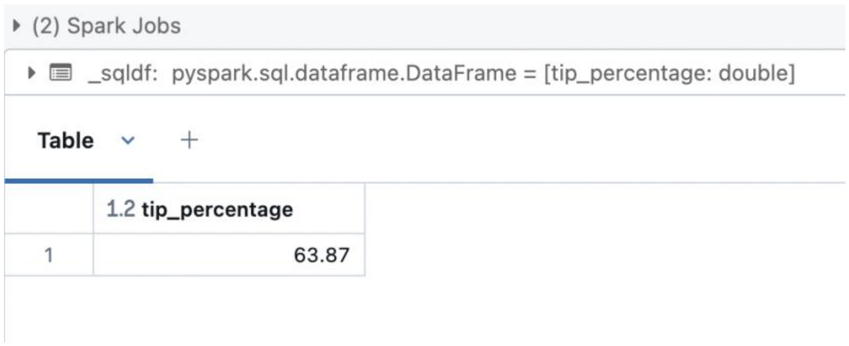


***Figure 13. Taxi trips trend by hour in Bronx***

The trip distance also changes depending on where the pickup_borough and dropoff_borough are, which is reflected in the taxi fare. These attributes will be included in the training of the machine learning model in future steps.

- Q4) Provide the percentage of trips where drivers received tips



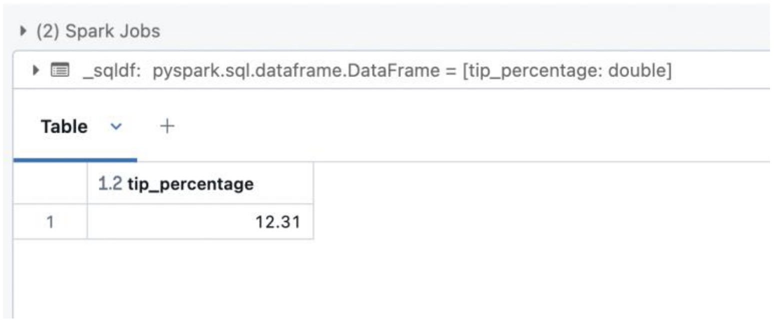
```
(2) Spark Jobs
_sqldf:  pyspark.sql.dataframe.DataFrame = [tip_percentage: double]
```

|  | 1.2 tip_percentage |
|---|---|
| 1 | 63.87 |

*Figure 14. Question 4 result*

- Figure 14 shows that 63.87% of trips include a tip.

- Q5) Provide the percentage of trips where drivers received tips of at least $5 in Q4 group.



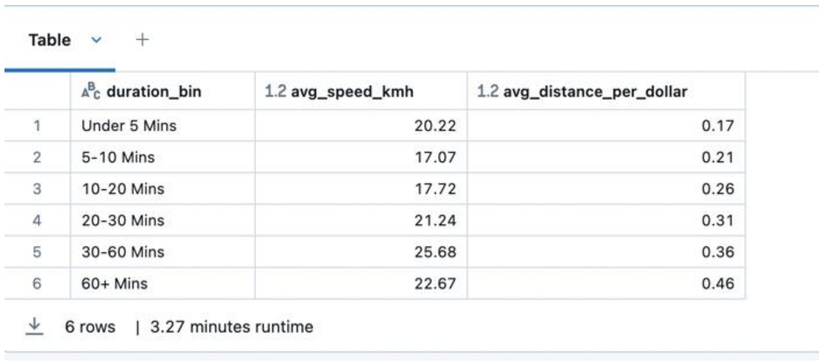
```
(2) Spark Jobs
_sqldf:  pyspark.sql.dataframe.DataFrame = [tip_percentage: double]
```

|  | 1.2 tip_percentage |
|---|---|
| 1 | 12.31 |

*Figure 15. Question 5 result*

- In Figure 15, it can be observed that 12.31% of taxi fares with tips are above $5.

- Q6) Classify average speed and distance in bins (Under 5mins, From 5 mins to 10 mins, From 10 mins to 20 mins, From 20 mins to 30 mins, From 30 mins to 60 mins, over 60mins)



Table

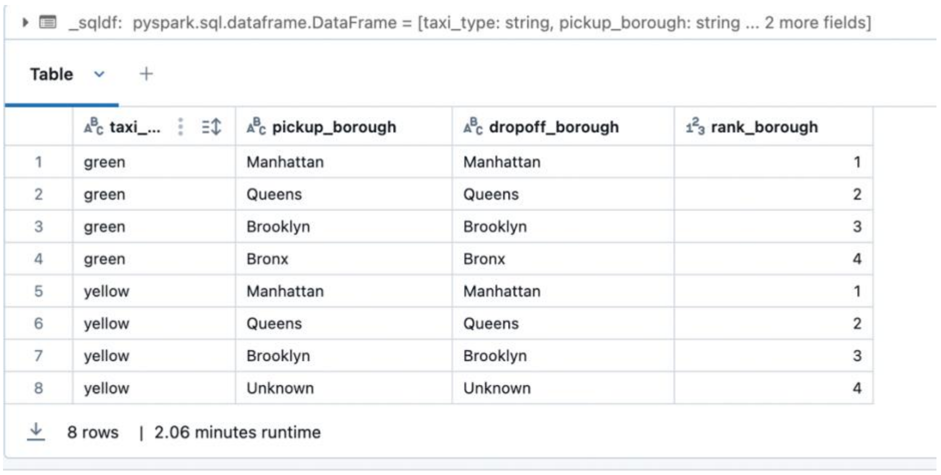
| | duration_bin | avg_speed_kmh | avg_distance_per_dollar |
|---|---|---|---|
| 1 | Under 5 Mins | 20.22 | 0.17 |
| 2 | 5-10 Mins | 17.07 | 0.21 |
| 3 | 10-20 Mins | 17.72 | 0.26 |
| 4 | 20-30 Mins | 21.24 | 0.31 |
| 5 | 30-60 Mins | 25.68 | 0.36 |
| 6 | 60+ Mins | 22.67 | 0.46 |

6 rows | 3.27 minutes runtime

*Figure 16. Question 6 result*

- Figure 16 shows the result (Classification of the average speed and distance per dollar in duration bins). Short trips (Under 5 Mins) have a relatively high average speed of 20.22 km/h but the shortest distance per dollar at 0.17 km per dollar. As the trip duration increases, the average distance per dollar also increases. Trips lasting 30-60 minutes have the highest average speed (25.68 km/h) and recorded a value per dollar at 0.36 km/dollar. For trips longer than 60 minutes, the average speed decreases slightly to 22.67 km/h, but the distance per dollar continues to increase, reaching 0.46 km/dollar.

- Q7) Provide the best duration bin for a taxi driver to maximise his income.



```
_sqldf: pyspark.sql.dataframe.DataFrame = [taxi_type: string, pickup_borough: string ... 2 more fields]
```

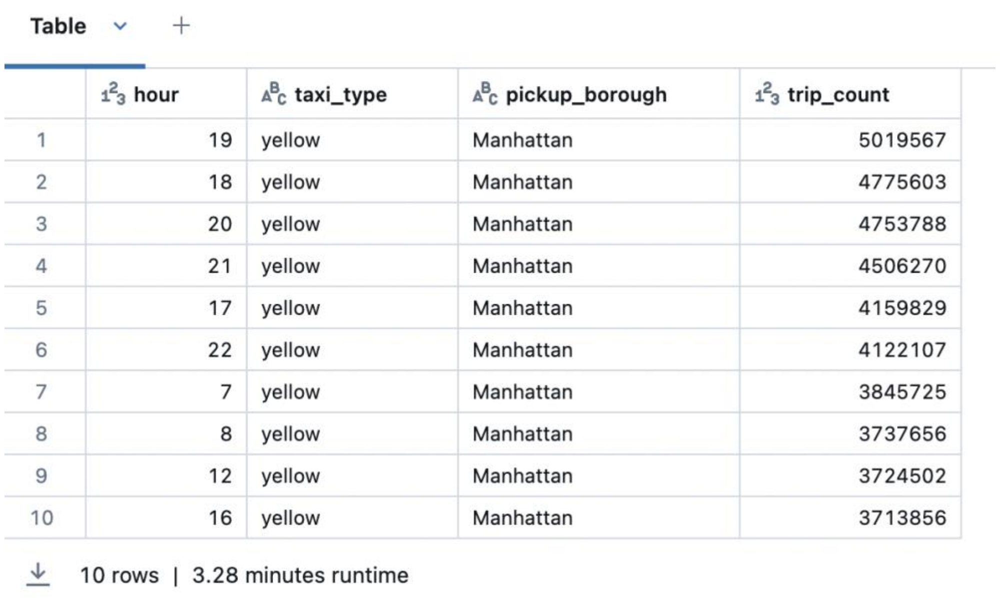
Table

| | taxi_... | pickup_borough | dropoff_borough | rank_borough |
|---|---|---|---|---|
| 1 | green | Manhattan | Manhattan | 1 |
| 2 | green | Queens | Queens | 2 |
| 3 | green | Brooklyn | Brooklyn | 3 |
| 4 | green | Bronx | Bronx | 4 |
| 5 | yellow | Manhattan | Manhattan | 1 |
| 6 | yellow | Queens | Queens | 2 |
| 7 | yellow | Brooklyn | Brooklyn | 3 |
| 8 | yellow | Unknown | Unknown | 4 |

8 rows | 2.06 minutes runtime

*Figure 17. Supporting information for question 7 (1)*

- Figure 17 illustrates the areas with the highest number of taxi trips when the trip duration is less than 5 minutes. It shows that the most frequent trips occur within Manhattan for both green and yellow taxis. The second most common trips are within Queens. Additionally, Figure 18 reveals that taxi usage peaks for both green and yellow taxis after 6 PM. To maximise driver income by minimising the distance per dollar during short trips, it is advised that both green and yellow taxi drivers focus on operating in Manhattan between 6 PM and 9 PM when taxi demand is highest for short trips.



| | hour | taxi_type | pickup_borough | trip_count |
|---|---|---|---|---|
| 1 | 19 | yellow | Manhattan | 5019567 |
| 2 | 18 | yellow | Manhattan | 4775603 |
| 3 | 20 | yellow | Manhattan | 4753788 |
| 4 | 21 | yellow | Manhattan | 4506270 |
| 5 | 17 | yellow | Manhattan | 4159829 |
| 6 | 22 | yellow | Manhattan | 4122107 |
| 7 | 7 | yellow | Manhattan | 3845725 |
| 8 | 8 | yellow | Manhattan | 3737656 |
| 9 | 12 | yellow | Manhattan | 3724502 |
| 10 | 16 | yellow | Manhattan | 3713856 |

10 rows | 3.28 minutes runtime

*Figure 18. Supporting information for question 7(2)*

## 6. Machine learning modelling (Part 3)

*Corresponding file: Part_3.ipynb, Part_3.html*

Apach Spark MLlib is a scalable machine learning library built on top of Apache Spark designed for large-scale data processing. It provides various algorithms and utilities for classification, regression, clustering, and collaborative filtering. To predict total amount of a trip, this project builds machine learning models using Spark MLlib. Figure 19 demonstrates the ML modelling process.

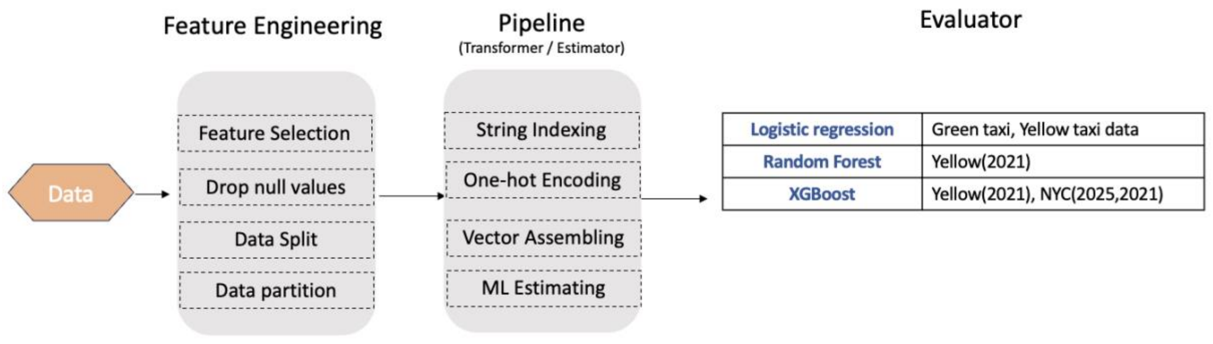


*Figure 19. Machine learning modelling process*

- Feature Engineering: To improve the model's performance and learning speed, unnecessary columns were dropped in the feature selection process. The table below is the columns that were selected and dropped. Moreover, based on the data analyses, we discovered that taxi trips vary by month, time, and day of the week, so we added these features. However, the year information was excluded to avoid bias, as certain years may have specific events or anomalies (e.g., pandemics, policy changes) that could skew the model's performance and generalizability.

| **Selected features** |
|---|
| ' DOLocationID', ‘PULocationID’, ' passenger_count’, trip_distance_km, 'extra', 'tip_amount', 'trip_duration', ‘taxi_type’, ‘month’, ‘hour’, ‘weekday’] |
| **Dropped features** |
| 'mta_tax', 'tolls_amount', 'store_and_fwd_flag', 'improvement_surcharge', 'RatecodeID' ,'congestion_surcharge', 'airport_fee', 'trip_distance', 'trip_type', 'ehail_fee', 'payment_type', 'VendorID', 'fare_amount’, ‘pickup_borough', 'pickup_zone', 'pickup_service_zone', 'dropoff_borough', 'dropoff_zone', 'dropoff_service_zone' |

- Data partitioning: Before the pipeline process, we specifically focused on the dataset from 2021, isolating it from the broader dataset that included records from 2015 to 2021. This choice ensured that our model captured the most recent trends and patterns. We applied a sampling technique to reduce the size of both the training and validation datasets by 10%, 50%. This approach was essential to manage computational resources efficiently during model experimentation while maintaining a representative subset of the original data. Repartitioning the data improves parallel processing capabilities during the training and validation phases. By distributing the data across ten partitions, we achieved more balanced workloads, minimising potential bottlenecks.

- Pipeline: By using Spark ML pipelines, we automate the sequence of data transformations, reducing the need for manual intervention. This pipeline includes String indexing, One-hot encoding, and Vector assembling. We conducted model training and evaluation manually to handle diverse datasets.

***Table 2. ML model evaluation (RMSE)***

| Model | Dataset | RMSE | | | |
|---|---|---|---|---|---|
| | | Baseline | Train | Validate | Test |
| Linear Regression | Green Taxi (8% train, 2% val) | 557.6 | 443.96 | 440.04 | - |
| | 2021 Yellow Taxi (8% train, 2% val) | 611.98 | 490.19 | 486.04 | 641.58 |
| XGB | 2021 Yellow Taxi (8% train, 2% val) | 611.98 | 349.23 | 359.77 | 520.54 |
| RandomForest | 2021 Yellow Taxi (8% train, 2% val) | 611.98 | 467.53 | 470.4 | 623.15 |
| XGB-1 | 2021 NYC Taxi (8% data) | 622.14 | 355.4 | - | 547.18 |
| XGB-2 | 2021 NYC Taxi (40% data) | 622.14 | 354.62 | - | 520.16 |
| XGB-2 | 2015 NYC Taxi (8% train, 2% val) | - | 339 | 331.43 | |

Linear Regression was used as a base Machine Learning model. The model exhibited reasonable RMSE values in the Green Taxi dataset. Train RMSE was 443.96, which is lower than the 557.6 baseline RMSE. However, it struggled with the Yellow Taxi data, text set RMSE(641.58) was higher than baseline RMSE(611.98), indicating a possible increase in complexity. Although the Random Forest model showed a firm fit to the training and validation sets, it failed to generalise well to the test set. On contrast, the **XGBoosting(XGB)** model consistently outperformed all other models, The significant reduction in RMSE values for both validation and test sets indicates its effectiveness in fare prediction. The model trained with 2021 data also produced a low RMSE (339) on 2015 data. This demonstrates that by training on the most recent data, it is possible to predict past data as well. As a result, this project's best model is XGB with lowest RMSE and processing time that was more than 10 minutes faster compared to other models. To robust this model, we applied hyperparameter tuning (max_depth and learning rate), which resulted in **max_depth = 7 and learning_rate = 0.3**. Deeper trees can help the model to capture more detailed patterns.

## 7. Conclusion

This project compromises evaluations of various regression models for NYC taxi fare prediction, revealing that XGBoost consistently delivers superior performance to traditional methods such as Linear Regression and Random Forest. Taxi companies can use trip amount predictions to implement dynamic pricing strategies based on demand, traffic conditions, and special events, helping maximize revenue. Furthermore, ride-hailing apps can provide users with estimated fares before they book a ride, improving customer satisfaction and decision-making. They can also be used to develop strategies for competitors.

## 8. Reference

- NYC Taxi and Limousine Commission (TLC) Documentations,

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

- Spark Documentations, https://spark.apache.org/mllib/