Big Data Processing on NYC taxi dataset with Databricks Spark

Big Data Engineering | Spring 2025

By Huy Duc Vu

Table of Contents

Project Overview

This big data project aims at analysing the New York Taxi and Limousine Commission (TLC) dataset, particularly New York City (NYC) famous yellow and green cabs, using Databricks Spark. The project focuses on large-scale data processing, cleaning, and transformation to prepare a reliable workflow of millions of taxi trip records for analysis. Business questions are then raised and answered through the utilisation of SparkSQL queries to provide insights into passenger demand, trip patterns, revenue distribution or customer behavior. Afterwards, predictive machine learning models are developed based on a clean dataset to estimate the total paid amount by the passengers, comparing with a baseline model derived from historical averages against much more sophisticated machine learning approaches, including regression models like ElasticNet and tree-based models like Random Forest.

This project includes some key stages:

1. Data ingestion, validation and cleaning of invalid records, and merging of separate yellow and green taxi parquet files with location lookup data.
2. SQL-based data analysis to answer particular business questions with context.
3. Machine learning models applied, in comparison with the baseline average-based model.

Data Source Analysis

This project uses public data from TLC regarding 2 major types of taxi in NYC, including green and yellow taxis, formatted in Parquet input files for Databricks ingestion with an additional CSV file containing a location lookup table.

1. Content and Structure

Details on the used datasets:

- Yellow taxi trips (parquet file)
    - Yellow taxis can legally pick up passengers via street-hail anywhere in NYC.
    - Includes data from 2009, when the trips were required to be recorded and provided to the TLC.

- Green taxi trips (parquet file)
    - Serves the areas underrepresented by the yellow cabs, complementing the service coverage.
    - Green cabs were introduced in NYC in August 2013, limited to outer boroughs and northern Manhattan.
- Taxi zone lookup table (csv file)
    - This data maps the numeric LocationID to detailed borough, zone and service zone in NYC.
    - This is essential for interpreting pickup and drop-off locations at the borough level.

2. Size and Format

This is a large-scale dataset including millions of rows of taxi records across years, particularly:

- Yellow taxi: 907,982,776 records
- Green taxi: 83,484,688 records
- Those data is stored in storage-efficient parquet format, but since it is too large to analyse directly in Pandas, Spark is required to process such files.

Although there are differences in terms of structure, the key fields are similar in both yellow and green datasets:

- Timestamp: `pickup_datetime`, `dropoff_datetime`.
- Locations: `PULocationID`, `DOLocationID`.
- Trip details: ratecodeid, vendorid, trip_type, trip_distance, derived trip duration and speed.
- Fare and payment records: `payment_type`, `fare_amount`, `extra`, `mta_tax`, `tip_amount`, `tolls_amount`, `improvement_surcharge`, `congestion_surcharge` and `total_amount`.
- Passenger information: `passenger_count`.

3. Data Quality

The input parquet files are raw, unstructured data, including multiple invalid records, raising certain issues such as:

- Trips with drop-off before or same as pickup datetime.
- Negative or zero values for distance, duration or fare.
- Unrealistic values: speed, duration, distance.
- Missing passenger counts or out-of-range values, since the maximum capacity of a taxi is 4 to 5 passengers.

For such errors in the dataset, the pre-processing steps consider:

- Ensure the data cleaning process removes invalid trips but keeps it aligned with the regulation of not removing over 10% of the total records.
- Cap or impute values for certain fields where dropping rows results in bias.
- Standardise the schema between the yellow and green datasets before union.

Overall Architecture/ Workflow

The project follows a structured pipeline to handle large-scale taxi records effectively, ensuring that each stage prepares the data for the following. The workflow is summarised as:

- Part 1: Data ingestion and preparation
    - Loaded the yellow and green taxi parquet files into Databricks Spark.
    - Imported the taxi zone lookup CSV file to enrich trips with borough and zone names.
    - Verified schema alignment and column consistency across the dataset before joining.
    - Applied filters to remove, replace or update unrealistic or invalid records.
    - Standardised schemas between yellow and green trips.
    - Joined 3 sources of data, including yellow taxi, green taxi and the location lookup table.
    - Saved as a Databricks Delta table for analysis and modelling in the upcoming stages.
- Part 2: Data Analysis based on Business questions

    - Created temporary views of the cleaned dataset for SQL exploration.
    - Extracted relevant records and logically formatted to answer the business questions.
    - Derived insights into driver strategies and efficiency.
- Part 3: Machine learning
    - Established a baseline model by using the average fare for certain criteria.
    - Performed feature engineering and feature selection.
    - Split data by date.
    - Built and evaluated predictive models to predict the total amount of the last 3 months of 2024.
    - Evaluated models using mainly root mean squared errors (RMSE), with reference to mean absolute errors (MAE) or R-squared.

Part 1. Data Ingestion and Preparation

Refer to code from: `Part 1.ipynb`

1. Objective

The first stage was to ingest the raw datasets, clean invalid records, and prepare a unified dataset for downstream analysis and modelling. This included handling both the yellow and green taxi parquet files and enriching them with the taxi zone lookup CSV.

2. Environment setup

- Throughout this project, the free edition of Databricks is used, along with the connection to free Serverless compute.
- Free Serverless configuration:
    - Memory: 32GB
    - Dependencies: scikit-learn, numpy, threadpoolctl version 3.2.0.

3. Data Ingestion Steps

Overall, the exploration and cleaning process between the yellow and green taxi dataset is similar.

- Data Loading

    - Generated a directory towards the file at catalog ‘workspace’, schema ‘bde’, and volume ‘assignment 2’.
    - Read the green and yellow taxi trip parquet files into Spark DataFrames (`df_green`, `df_yellow`).
    - Imported `taxi_zone_lookup.csv` to map `PULocationID` and `DOLocationID` under `df_lookup`.
- Temporary views for SQL exploration
    - Created SQL views (`df_green`, `df_yellow`, `df_lookup`) for easier exploration and queries since we utilised SQL in big data for simplicity.
- SQL queries to explore the dataset
    - Flag records with drop-off before or the same as pickup datetime, or records that are before the valid start time for each taxi (2013 for green taxi and 2009 for yellow taxi).
    - Derive speed in miles per hour and identify out-of-range data.
    - Select the passenger count in descending order to identify irrelevant records.
    - Identify out-of-range `trip_distance`, and `total_amount`.
    - Calculate duration in minutes to define out-of-range data.
    - Check for `PULocationID` and `DOLocation` that do not match any value in the location lookup table.
    - Identify invalid records which does not match the predefined dictionary in categorical data, including `ratecodeid`, `vendorid`, `payment_type`, `store_and_fwd_flag`, and `trip_type`.
- Cleaning rules applied to filter the dataset
    - Date sanity check: kept trips within a defined date range (2013 for green cab and 2009 for yellow cab – based on their establishment and the start date of reporting to the TLC).
    - Chronological order: excluded trips where `dropoff_dt` is before `pickup_dt`.

- Trip duration: capped between 1 minute and 150 minutes (since shorter than 1 minute seems to be an error, while longer than 150 minutes trips are unrealistic).
- Trip distance: capped between 0.3 miles (~0.5 km) and 200 miles (since distances under 0.3 miles or over 200 miles are not realistic in NYC context).
- Speed filter: calculated `trip_distance / duration_hr` and excluded trips with speeds <1 mph or >90 mph (this limit is based on the NYC regulation since the average distance lower than 1mph and higher than 90mph looks like errors as the NYC general speed limit is 25mph).
- Fare amount: excluded negative, null fares or unrealistic fare amounts that are over $200 (including tips).
- Passenger count: since most of the invalid passenger count records have valid information in other fields such as distance, duration or total fare amount, removing all the invalids seems to be unreasonable. It is more practical to cast missing and 0 count to 1 (since 0 is close to 1, which the drivers may mistakenly enter the data), while passenger count above 5 is marked as 5 (this is the maximum number of allowed passengers in a regular taxi – 4 adults and possibly 1 child)
- Vendor ID: Some recorded vendor IDs are not in the dictionary; therefore, it is more reasonable to transform those data into NULL values (for green taxi, there is only 1 invalid type of vendor ID, so it is transformed into the closest type).
- Payment type and Ratecode ID: These fields have a specific type for unknown data; hence, it is more reasonable to transform null values into the unknown field.

- Data merge between datasets:
  - Since the schemas of datasets are not exactly matched, only similar fields are kept, except for `trip_type`, where all yellow data is cast as street-hail type.
- Final table: The final table is saved as `final_trips_cleaned`.

4. Data summary after cleaning

- Count of the dataset records after cleaning
    - Green taxi: 80,458,710 records (96% of the original data).
    - Yellow taxi: 889,880,080 records (98% of the original data).
- Count of dataset records after merge: 970,457,848 rows.

Part 2. Data Analysis for Business Questions

Refer to code from: `Part 2.ipynb`

1. Objective

The objective of this part is to answer a series of business-oriented questions about taxi trip patterns, passenger demand, and revenue distribution using SQL queries on the cleaned dataset. This stage focused on deriving actionable insights such as identifying the busiest time periods, comparing yellow vs green taxi operations, analysing borough-to-borough flows, and understanding tipping behaviours. These findings provide a data-driven foundation for evaluating taxi service efficiency and guiding driver strategies.

2. Question 1

- Code logic: Group trips by year-month, calculate total trips, average passengers, average fare, busiest day of week, and busiest hour.
- Insight: Monthly demand shows strong evening and weekend peaks, with Friday evenings being consistently the busiest. Most of the time, passengers remain around 1 to 2, while revenue per trip is stable across months.

	year_month	total_trips	busiest_dow	busiest_hr	average_passenger	average_amount_per_trip	average_amount_per_passenger
1	2009-01	957	Thursday	0	2	18.58	11.43
2	2011-01	2	Monday	23	1	12.8	12.8
3	2011-02	1	Tuesday	0	1	15.8	15.8
4	2014-01	14353935	Friday	19	2	14.4	8.69
5	2014-02	13844429	Saturday	19	2	14.53	8.83
6	2014-03	16461114	Saturday	19	2	14.69	8.95
7	2014-04	15696588	Wednesday	19	2	14.98	9.11
8	2014-05	15962270	Friday	19	2	15.55	9.45
9	2014-06	14928168	Sunday	19	2	15.56	9.45
10	2014-07	14170409	Thursday	19	2	15.21	9.24
11	2014-08	13654679	Friday	19	2	15.36	9.33
12	2014-09	14502851	Tuesday	19	2	15.6	9.59
13	2014-10	15562781	Friday	19	2	15.49	9.53
14	2014-11	14612044	Saturday	19	2	15.33	9.41
15	2014-12	14502352	Wednesday	19	2	15.39	9.44
16	2015-01	14025818	Saturday	19	2	14.71	9.08
17	2015-02	13793363	Saturday	19	2	15.24	9.48
18	2015-03	14818278	Sunday	19	2	15.59	9.7
19	2015-04	14492430	Thursday	19	2	15.82	9.81
20	2015-05	14695085	Saturday	19	2	16.26	10.03

3. Question 2

- Code logic: Calculate average, median, min, and max for duration (minutes), distance (km), and speed (km/h) per taxi colour.
- Insight: Yellow taxis generally have longer trips and higher maximum speeds compared to green taxis, which reflects their focus on Manhattan and airport trips. On the other hand, green taxis serve only shorter, borough-level trips.

	color	avg_duration_min	median_duration...	min_duration_min	max_duration_min	avg_distance_km	median_distance_km
1	green	13.84	10.72	1	120	4.93	3.19
2	yellow	14.46	11.35	1	120	4.93	2.78

min_distance_km	max_distance_km	avg_speed_kmh	median_speed_kmh	min_speed_kmh	max_speed_kmh
0.34	148.62	20.55	18.61	1.61	112.65
0.34	160.77	18.95	16.62	1.61	112.65

4. Question 3

- Code logic: Aggregate trips by `colour`, `PU borough`, `DO borough`, `month`, `day-of-week (dow)`, `hour`, `reporting trips`, `avg distance`, `avg fare`, `and revenue`.
- Insight: The highest-revenue flows are Manhattan and airports (JFK, LaGuardia). Borough-internal trips (same pickup/dropoff borough) contribute high volumes but lower average fares.

	color	PU_Borough	DR_Borough	year_month	dow	hr	total_trip	avg_distance_km	average_amount_per_trip	total_revenue
1	green	Bronx	Bronx	2014-01	Friday	0	187	3.7	10.13	1894.98
2	green	Bronx	Bronx	2014-01	Friday	1	142	3.39	9.61	1365.1
3	green	Bronx	Bronx	2014-01	Friday	2	82	3.94	10.33	847.25
4	green	Bronx	Bronx	2014-01	Friday	3	62	3.46	9.82	608.56
5	green	Bronx	Bronx	2014-01	Friday	4	56	3.56	10.14	568
6	green	Bronx	Bronx	2014-01	Friday	5	68	3.67	9.81	667
7	green	Bronx	Bronx	2014-01	Friday	6	129	4.02	10.09	1301.1
8	green	Bronx	Bronx	2014-01	Friday	7	493	3.53	10.18	5017.1
9	green	Bronx	Bronx	2014-01	Friday	8	808	3.37	10.17	8220.29
10	green	Bronx	Bronx	2014-01	Friday	9	577	3.46	9.96	5745.19
11	green	Bronx	Bronx	2014-01	Friday	10	556	3.76	10.63	5910.82
12	green	Bronx	Bronx	2014-01	Friday	11	550	3.57	10.17	5594.41
13	green	Bronx	Bronx	2014-01	Friday	12	577	3.6	10.26	5917.86
14	green	Bronx	Bronx	2014-01	Friday	13	573	3.68	10.75	6162.1
15	green	Bronx	Bronx	2014-01	Friday	14	663	3.56	10.47	6939.97
16	green	Bronx	Bronx	2014-01	Friday	15	743	3.66	10.9	8096.91
17	green	Bronx	Bronx	2014-01	Friday	16	780	3.51	11.52	8986.65
18	green	Bronx	Bronx	2014-01	Friday	17	858	3.39	11.53	9889.92
19	green	Bronx	Bronx	2014-01	Friday	18	866	3.47	11.2	9696.27
20	green	Bronx	Bronx	2014-01	Friday	19	734	3.68	11.14	8175.72

10,000+ rows | Truncated data

5. Question 4

- Code logic: Rank borough pairs in 2024 by total revenue and compute revenue share vs overall.
- Insight: The top 10 flows are dominated by Manhattan internal trips, together contributing a majority share of annual revenue. This highlights the economic importance of short-distance intra-borough trips.

	rank	PU_Borough	DR_Borough	total_revenue_2024	total_revenue_share_percentage_2024
1	1	Manhattan	Manhattan	712834994.23	62.83
2	2	Queens	Manhattan	173793871.19	15.32
3	3	Manhattan	Queens	74532940.27	6.57
4	4	Queens	Brooklyn	38704177.69	3.41
5	5	Manhattan	Brooklyn	32927141.95	2.9
6	6	Queens	Queens	32924799.83	2.9
7	7	Manhattan	EWR	12532145.81	1.1
8	8	Queens	Unknown	9722054.02	0.86
9	9	Brooklyn	Brooklyn	7811496.16	0.69
10	10	Brooklyn	Manhattan	7720958.5	0.68

6. Question 5

- Code logic: Count trips where tip_amount > 0 vs all trips.
- Insight: Roughly 63% of trips included tips, showing tipping is a common practice, especially in yellow taxis, where card payments dominate. The other 37% seems to tip with cash since cash tips are not recorded in the system.

	trip_with_tips	total_trip	pct_trip_with_tips
1	612049272	970338790	63.08

7. Question 6

- Code logic: Filter trips with tips, then calculate the share with tip_amount >= 15.
- Insight: Only about 5–8% of tipped trips reached $15 or more; most of them are likely from long-distance rides since short rides account for low tips, which is in accordance with the low total paid amount.

	trip_with_tips	trip_tips_15_or_more	pct_trip_with_tips_15_or_more
1	612049272	4980039	0.81

8. Question 7

- Code logic: Bucket trips into bins (<5, 5–10, 10–20, 20–30, 30–60, 60+ minutes), calculate average speed and distance per dollar.
- Insight: 30–60 min trips balanced efficiency and profitability, while short trips (<5 min) were inefficient (low distance per $). Very long trips (>60 min) gave the best efficiency but were less frequent.

	duration_bin	avg_speed_kmh	avg_km_per_dollar
1	<5	20.03	0.17
2	5-10	17.22	0.21
3	10-20	17.92	0.26
4	20-30	21.39	0.31
5	30-60	25.8	0.38
6	>60	22.68	0.56

9. Question 8

- For short trips (<5 mins), it shows very poor efficiency (only 0.16 km per $), which is likely due to base fare dominating.
- For medium trips (20–60 mins), this shows a balance between higher speed and steadily improving efficiency.
- For long trips (>60 mins), this is the best efficiency (0.56 km per $), but risky because:
    - Fewer available rides per shift (one long trip vs many short ones).
    - High idle time returning from long-distance drop-offs.

Therefore, drivers should aim for 30–60 minute trips, which maximise speed and income stability, without the risks of downtime that come with very short or very long trips.

Part 3. Machine Learning

Refer to code from: `Part 3.ipynb`

1. Objective

The goal of the machine learning part is to build predictive models for the trip total amount and compare them against a baseline model derived from Part 2 – Q3c (average total per colour, borough pair and time). By evaluating different machine learning approaches, we aimed to determine whether predictive modelling could improve fare estimation and provide deeper insight into the factors driving trip prices.

2. Baseline model

The baseline model performance is derived from Part 2 – Q3c, to which each trip's predicted fare is equal to the average fare for its group (`color`, `PU_Borough`, `DR_Borough`, `month`, `dow`, and `hour`).

- Performance: RMSE = 8.17
- This serves as a simple benchmark, which captures broad temporal and spatial patterns but ignores trip-specific details like distance or duration.

3. Model Selection and Feature Engineering

Due to the limitation in computing resources of the free Databricks edition, only a minimal number of fields are selected for the machine learning models.

- All fare-related fields except for total_amount are removed due to the potential of data leakage, as they are components of the total_amount.
- Categorical fields like `passenger_count`, `trip_type`, `ratecodeid` or `vendorid` are also removed since they have little predictive power towards the target.

Besides, some related fields are generated to contribute to the predictive power of the target:

- `duration_min`: the duration of the trips in minutes.
- `same_borough`: identify if the boroughs are the same between the pickup and drop-off location.

Notably, since the data includes around 900 million records, it is essential to filter the data only from 2023, with a sample of 10% to run through machine learning models. This is reasonable since the taxi fares are monetary unit, which suffers from inflation, and the machine learning model should only be trained using recent years' data.

4. Model 1: ElasticNet Regression

- Features: trip_distance, duration_min, speed_mph, same_borough, categorical fields (`color`, `PU_Borough`, `DR_Borough`, `payment_type`).
- Approach:
    - Use OneHotEncoder + StandardScaler.
    - Apply ElasticNet to combine L1 and L2 regularisation → balances interpretability and robustness.
    - The dataset is capped to recent years and sampled due to memory constraints.
- Results: RMSE = 6.646
- Insight:
    - The RMSE is improved compared to the baseline, which is a good sign.
    - Regularisation handles multicollinearity between distance, duration, and speed, which provides interpretable coefficients for feature influence.
    - Strengths: faster training, lightweight, and interpretable coefficients.
    - Weakness: may be limited in capturing non-linear interactions between features.

5. Model 2: Random Forest

- Features: same as ElasticNet, but used OrdinalEncoder instead of OHE (to avoid exploding feature space).
- Approach:
    - An ensemble of decision trees to capture non-linear relationships and feature interactions.

    - Parameter tuning simplified (limited estimators, capped depth) due to Databricks resource limits.
- Results: RMSE = 5.134
- Insight:
    - Achieve the lowest RMSE among models (reduction of nearly 22.8% compared to ElasticNet), showing that tree ensembles better capture complex fare structures (airport surcharges, borough effects).
    - Feature importances highlight distance and duration as dominant predictors, with boroughs and payment_type adding secondary effects.
    - Strengths: significantly better performance, handles non-linearities, and identifies feature importances.
    - Weakness: slower training and more resource-intensive (hundreds of decision trees).

6. Best model selection

The Random Forest is chosen as the best-performing model for this project because:

- Performance: Achieve the lowest RMSE (5.134), showing a clear improvement over both ElasticNet (6.646) and the baseline. This means its fare predictions are closer to actual observed amounts.
- Complexity handling: Taxi fares involve many non-linear patterns such as borough combinations, airport surcharges, and traffic-influenced durations. Random Forest captures these interactions automatically, which linear models struggle with.
- Robustness: Works well without heavy preprocessing; less sensitive to outliers and skewed distributions.
- Interpretability (relative): While not as transparent as ElasticNet, feature importances clearly highlight distance, duration, and boroughs as key drivers, providing useful business insights.
- However, the trade-off is the processing time, which is much longer than Elastic Net, which makes it appealing for large datasets.

For such reasons, despite higher compute cost, Random Forest is the most suitable model for this project because it substantially improves predictive accuracy while remaining interpretable enough to derive business insights.

Potential Issues

Working with a dataset of this scale (nearly 900M rows) in a limited Databricks free-tier SQL warehouse introduced several challenges. Key issues and how they were resolved:

- Dataset size & memory limits
    - Issue: `.toPandas()` on full data caused memory errors and warehouse crashes.
    - Solution: Restrict the dataset to recent years and apply sampling before running through machine learning models.
- Model training is too slow with GridSearchCV
    - Issue: Running hyperparameter tuning (over 30 fits) on ElasticNet and Random Forest models took over an hour and sometimes did not finish.
    - Solution: Reduced grid size, switched to RandomizedSearchCV with 4 iterations to ensure timely results.

Conclusion

This project successfully demonstrates how large-scale NYC taxi trip data can be cleaned, analysed, and modelled using Databricks and scikit-learn. Part 1 produces a unified, validated dataset; Part 2 answers business questions on trip demand, revenue flows, and tipping behaviour; and Part 3 builds predictive models for fares. While the baseline provided a simple benchmark, the Random Forest achieved the best accuracy (RMSE = 5.134), showing the value of non-linear modelling. Despite computational limits, careful sampling and simplified tuning enabled meaningful insights and reliable predictions.