Data Processing and Analysis of New York City Taxi Using Databricks Spark

Phuong A. Pham

Department of Transdisciplinary Innovation, University of Technology Sydney

94693: Big Data Engineering

William So

October, 2024

Table of Contents

SECTION 1.
Introduction

1.1 Project overview

The project focuses on data processing and analysis using Databricks Spark, with the primary goal of leveraging Apache Spark to conduct a comprehensive analysis of a high-volume dataset. It aims to analyse a large dataset from New York City taxi trips by loading, transforming, and performing detailed analysis to derive valuable insights and predictions. One of the key aspects of the project is the efficient handling of data stored in Parquet and CSV formats, with a particular emphasis on the Parquet file format due to its superior performance and efficiency in big data processing. Throughout the project, Databricks is utilised as a core platform to process and manipulate data, with the dataset stored in Microsoft Azure Blob Storage. Additionally, various tools and technologies are employed to process and transform data, including Python, PySpark, and Spark SQL. After being ingested from Azure, the dataset undergoes extensive cleaning to remove unrealistic records, then be explored using Spark SQL to extract insights into taxi operations, trip patterns, and passenger behaviour. By the end of the project, Spark ML pipelines will be used to build and train predictive models, with performance evaluated against a baseline to ensure accuracy in predicting trip totals.

1.2 Project objectives

The primary goal of the project is to conduct a comprehensive analysis of a large dataset using Apache Spark, with a focus on data ingestion, transformation, machine learning model development for predicting profound findings.

To support the project goal, we have the following objectives:

- Ingest large datasets from multiple parquet files and a csv file using Databricks and Azure Blob Storage.
- Explore and clean data to remove unrealistic trips
- Conduct data transformation
- Execute SQL query to answer to key business questions
- Identify important and meaningful features which can improve the machine learning models' performance
- Select appropriate machine learning models and indicate the reasons of choice.
- Build and train models using Spark ML pipelines and baseline model to predict total amount of a trip.
- Identify issues/bugs encountered throughout the project and suggestion solutions to solve these issues.

1.3 Project workflow

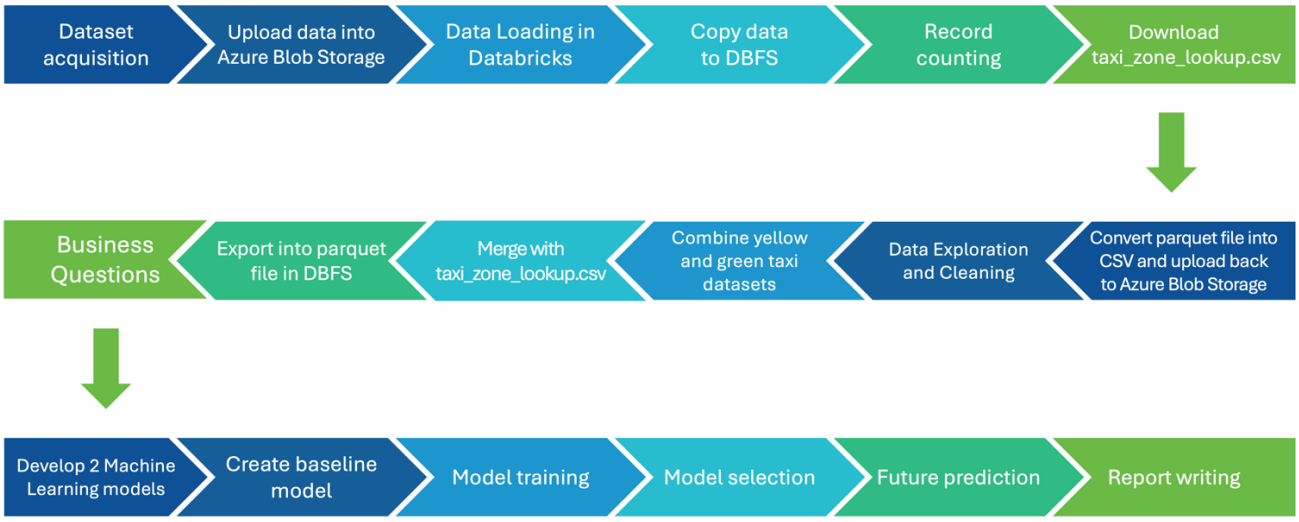

Figure 1: Project Workflow

1.4 Tools used

The project used various tools and technologies to facilitate data processing, analysis, and model development, including:

- Microsoft Azure: In this project, Azure Blob Storage was used as the primary storage solution for the taxi datasets. The data was securely uploaded to Azure Blob Storage, which then allows the data analyst to easily accessed for further analysis.
- Databricks: This is the primary platform of the project for large-scale data processing and analytics. Its integration with Microsoft Azure allowed the data analyst to easily mount the Blob Storage, enabling seamless access to the datasets.
- Apache Spark: This is a powerful open-source computing system that provides an interface for programming entire clusters.
- Python: Python is utilized in the project for data cleaning, transformation, analysis, model development.
- Pyspark: In this project, PySpark was used for data manipulation, transformation, and machine learning model development,
- SparkSQL: In this project, Spark SQL was used for executing SQL queries on the datasets, answering business questions

SECTION 2.
Data Understanding

The dataset for this project is provided by the New York City Taxi and Limousine Commission (TLC), a company which has been responsible for managing license and regulating New York City's taxis since 1971. The dataset comprises 16 parquet files, representing taxi data from 2015 to 2022 of two types of taxis cabs: yellow and green taxis, each with distinct operational characteristics.

Yellow taxi cabs, which have been provided to TLC since 2009, are famous vehicles in New York City. There are approximately 13,600 licensed yellow taxis in the city, which are permitted to pick up passengers via street hails all five boroughs (Queens, Manhattan, Brooklyn, Bronx, and Staten Island).
The green taxi service, which first began operating in August 2013, with the purpose of improving taxi availability in certain designated areas.

Each parquet file contains detailed trip-level data, including:

- Pick-up and drop-off dates/times: Time stamps of when and where passengers began and ended their journeys.
- Pick-up and drop-off locations: location in which the taximeter was engaged/disengaged.
- Passenger counts: The number of passengers reported by the taxi driver for each trip.
- Trip distance: The elapsed trip distance in miles reported by the taximeter.
- Rate Code ID: 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 forward flag: shows whether the trip record was held in vehicle memory or not
- Payment type: indicates how the passenger paid for the trip.
- Fare amount
- Itemized fares: including information about fare amount, taxes, tip amount, and surcharges.

Additionally, the dataset includes a `taxi_zone_lookup.csv` file, which provides geographic information related to the trips. This file maps locations to specific taxi zones, boroughs, and service areas, enabling spatial analysis of the trips and helping categorize data by borough, zone, or service area.

SECTION 3.
Data Ingestion and Preparation

3.1 Setup and data loading into Azure Blob Storage

Step 1: Create a Storage Account called `newyorktaxi`

Step 2: In `newyorktaxi` storage account, I created a container called `taxidata` and uploaded 16 parquet files into the new container.

Step 3: Retrieve *Storage account name* and *Storage account access key*

1. Go to your Azure Storage account `newyorktaxi`.
2. Navigate to the *Access keys* under *Security + networking* section.
3. Click on Show button next to *Key*, and copy *Key code*

3.2 Ingesting data into Databricks

Step 1: Create a compute cluster

1. Navigate to the Compute in the left sidebar
2. Click on Create Compute button

Step 2: Open Databricks Workspace

Step 3: Launch a Databricks Notebook and rename it to 'Assignment 2'

Step 4: Define the storage account details using the credentials getting from step 2, including storage account name, access key, and container name.

Step 5: Mount the Blob Storage container. The command below mounts the `taxidata` container from the `newyorktaxi` storage account as a file system at `/mnt/taxidata`.

3.3 Dataset record count verification

The total number of rows for yellow and green dataset were counted by reading the files stored on DBFS. The results are as follows:

Green taxi: 66,200,401

Yellow taxi: 663,055,251

3.4 Data format optimization: Parquet and CSV

After converting the "Green" 2015 parquet into a csv file and send it to Azure Blob Storage, the size of file format are as follows:

- CSV file size: 2.03 GiB
- Parquet file size: 385.81 MiB

The significant size difference between the Parquet and CSV files highlights the efficiency of the Parquet format.

Table 3: Comparison of CSV and Parquet file formats

Feature	CSV	Parquet
File size	2.03 GiB	385.81 MiB
Storage format	Row-based	Columnar (More efficient data access and compression)
Data type	All values stored as strings	Supports complex data types and optimized storage
Data retrieval	Reads entire row (even unused columns)	Reads only required columns, more convenient when dealing with large dataset
Performance	Slower	Faster for analytical queries
Data Consistency	No built-in mechanisms	Includes metadata for data consistency
Optimized for analytics	Not optimized for analytical processing	Specifically designed for analytical workloads

SECTION 4.
Data Transformation and Manipulation

4.1 Remove trips finishing before the starting time

One necessary step in the data cleaning process was ensuring that all recorded taxi trips had valid time sequences, specifically that the drop-off time occurred after the pickup time. Any trips where the drop-off timestamp is recorded earlier than the pickup timestamp are logically inconsistent. To handle this issue, the following steps are applied:

- Convert datetime to timestamp: The pickup and drop-off datetime were initially in string format in both the yellow and green taxi datasets. These features were converted to timestamp format using `to_timepstamp()` function.
- A filter was applied to remove any trips where drop-off time occurred before pickup time.

4.2 Remove trips where the pickup/dropoff datetime is outside of the range

4.3 Remove trips with negative speed

To handle this cleaning step, I first converted pickup and drop-off datetime to UNIX timestamp format. The reason for this transformation would be that UNIX timestamp represent time as the number of seconds, which make it easier for computing. This package was imported using the code: `from pyspark.sql.functions import unix_timestamp`.

- A new column called `trip_duration` was generated in both datasets by calculating the difference between `dropoff_timestamp` and `pickup_timestamp`.
- After calculating the trip duration, the filter function is applied to keep only those trips where the `trip_duration` is greater than 0.
- Speed is calculated by dividing the distance by the trip duration.
- A filter was applied to remove any trips which has negative speed.

4.4 Remove trips with very high speed

Removing trips with speeds exceeding the limit is essential to enhance the quality of the dataset, as this step would ensure that the dataset reflects more realistic driving conditions and aligns with safety regulations. Within New York City, the limit speed limit is 55 mph.
I first defined a variable called `speed_limit` and assign it a value of 55 km/h.
The `filter()` function is used separately in both `df_green` and `df_yellow` dataset to retain only those rows where the calculated speed of the taxi is less than or equal to the defined `speed_limit`.

4.5 Remove trips that are travelling too short or too long (duration wise)

Both extremely short and extremely long trips may lead to data errors since it can influence average trip duration, trip distance predictions, etc. Therefore, removing trips which have

too long or too short duration is an important cleaning step. To handle this issue, the following steps are applied:

- I first checked the minimum and maximum duration of two datasets, then I counted the number of trips which were longer than 7200-second drive to ensure not to remove a huge amount of dataset. The result showed that only 370 thousand trips were recorded to last longer than 7200 seconds (2 hours) out of a total of over 66 million trips, hence, setting the duration limits of 2 hours may not have a significant impact on the quality of dataset.
- Define the minimum trip duration as 180 seconds (3 minutes) and define the maximum trip duration as 7200 seconds (2 hours). Any trip outside this range will be removed.
- The `filter()` function is used separately in both `df_green` and `df_yellow` dataset to retain only those rows where the real trip duration is longer or equal than the minimum, and less or equal than the maximum trip duration.

4.6 Remove trips that are travelling too short or too long (distance wise)

Similarly, trips which were too long or too short in terms of distance were removed using the code below

- Define the minimum distance for a trip as 0.5 mph, and define the maximum distance as 50mph. Any trip outside this range will be removed.
- The `filter()` function is used separately in both `df_green` and `df_yellow` dataset.

4.7 Remove trips that have invalid number of passengers

According to NYC Taxi and Limousine Commission regulations, the maximum number of passengers allowed in a taxicab is 5 passengers. Therefore, all trips having the number of passengers outside this range will be considered invalid records.

Table 4: Number of removed records after each step

	Number of removed records	
	Green	Yellow
Trips finishing before the starting time	838	81,067
Trips where the pickup/dropoff datetime is outside of the range	793	3,192
Trips with negative speed	75,129	645,276
Trips with very high speed	149,244	703,631
Trips that are travelling too short or too long (duration wise)	4,247,231	33,383,113
Trips that are travelling too short or too long (distance wise)	1,164,131	13,082,541
Trips that have invalid number of passengers	2,735,145	25,344,371

After filtering, the green taxi dataset has 57,827,890 records, while the yellow taxi dataset has 589,812, 060 records.

4.8 Combine the yellow and green taxi

Since schema of the two datasets are not the same, some steps were handled to make 2 datasets consistent.

- I dropped column `ehail_fee` from `df_green` because most of records in this dataset are null values.
- A new column named `taxi_color` is added to separated green and yellow taxi cabs later in part 2.
- `.unionByName()` function was used to combine two datasets.

4.9 Combine with location data

The combined data was then combined with the location data on two location columns: pickup location (PUBorough), and drop-off location (DOBorough).

4.10 Export

Data frame was exported as a parquet file named `nyc_taxi_data.parquet` in the specified DBFS location, overwriting any existing file with the same name.

SECTION 5.
Business Questions and Analysis

In this section, we address several key business questions utilising Spark SQL to provide insights into taxi trip patterns, passenger behaviour, etc.
To use Spark SQL queries, temporary view is required. `createOrReplaceTempView()` was used to create a **data_table** from the combined dataframe of previous step. I also created **green_table** and **yellow_table** to handle business question 2 and 3, which require statistics of each taxi colour (yellow and green).

Question 1: Yearly and monthly trip analysis

For each year and month:

(a) What was the total number of trips?
(b) Which day of the week (e.g., Monday, Tuesday) had the most trips?
(c) Which hour of the day had the most trips?
(d) What was the average number of passengers per trip?
(e) What was the average amount paid per trip?
(f) What was the average amount paid per passenger?

- Year_month information was extracted using `DATE_FORMAT()` and The format string `'yyyy-MM'`.
- The total number of trips was calculated using the `COUNT()` function, which counts all entries in the dataset.
- The day of the week with the most trips was determined by using the `COUNT()` function, which counts the total number of trips for each day of the week within each year and month. The `DATE_FORMAT()` function is utilized to extract the day of the week (Monday, Tuesday, etc.) from the pickup datetime, and '`EEEE`' format will return the full name of the day.
- The hour of the day with the most trips was calculated using the `HOUR()` function to extract the hour from the pickup datetime and the `COUNT()` function to aggregate the total trips for each hour.
- The average number of passengers for each year and month was calculated using the `AVG()` function applied to the *passenger_count* column.
- The average amount paid per trip was computed using the `AVG()` function on the *total_amount* column, allowing us to assess the average fare collected from passengers per trip.
- The average amount paid per passenger was calculated by dividing the total_amount by passenger_count for each trip and then applying the AVG() function to find the overall average for each year and month.

	year_month	total_trips	day_of_week	hour_of_day	avg_passenger	avg_total_amount	avg_amount_per_passenger
1	2015-01	157287	Friday	19	1.5325106334280647	15.098523145596491	12.567613943518635
2	2015-01	152044	Saturday	19	1.5889808213411907	13.72140946042661	11.140416543898507
3	2015-01	147997	Saturday	23	1.6077555626127558	15.241422866687769	12.290633949791301
4	2015-01	147771	Friday	18	1.5138626658816683	15.331735523216713	12.831006383756442
5	2015-01	146085	Saturday	18	1.5714686655029606	13.624205907526372	11.107389292766179
6	2015-01	145233	Friday	20	1.5390854695558172	14.861050381119458	12.322833925489471
7	2015-01	145168	Friday	22	1.585487159704618	15.379618373203392	12.531301147640267
8	2015-01	143237	Friday	23	1.581330242884171	15.879558633601484	12.968335024008919
9	2015-01	143077	Saturday	22	1.6088190275166518	14.806905372638795	11.919402688765228
10	2015-01	139981	Friday	21	1.55663268586451	15.104676134625588	12.436422288266728
11	2015-01	138089	Thursday	19	1.5040589764572123	15.139120567177883	12.765395251134525
12	2015-01	134565	Thursday	18	1.5051462118678705	15.361560658425075	12.943492979852584
13	2015-01	131801	Saturday	0	1.5820972526763832	15.94155507166797	13.006065799959313
14	2015-01	131542	Saturday	20	1.582703623177388	14.364021833336007	11.699890645828392
15	2015-01	131272	Thursday	20	1.5061170699006643	15.322964760199559	12.926395088065314
16	2015-01	130288	Thursday	21	1.5239776495149209	15.936710211231889	13.34828629395818
17	2015-01	130217	Saturday	21	1.5949069629925432	14.430731240933687	11.709531103211548
18	2015-01	126541	Saturday	17	1.5609723330778167	13.867726586644428	11.324725335399988
19	2015-01	123409	Thursday	22	1.5308688993509387	16.23152930500106	13.54326314666788
20	2015-01	122686	Saturday	15	1.5514321112433367	13.987043590960331	11.498984669536886

Figure 5.1: Screenshots of business question 1 result

Question 2:

For each taxi color (yellow and green):

(a) What were the average, median, minimum, and maximum trip durations in minutes?

(b) What were the average, median, minimum, and maximum trip distances in kilometers?

(c) What were the average, median, minimum, and maximum speeds in kilometers per hour?

For each taxi color (yellow and green):

- The average trip duration, average trip distance, and average speed for each taxi colour were calculated using the `AVG()` function. The results were rounded to two decimal places using the `ROUND()` function.
- The median trip duration, median trip distance, and median speed were calculated using the `PERCENTILE()` function with 0.5 as the percentile value. The result was rounded to two decimal places.
- The minimum trip duration, minimum trip distance, and minimum speed were calculated using the `MIN()` function.
- The maximum trip duration, maximum trip distance, and maximum speed were calculated using the `MAX()` function.
- The trip duration, originally in seconds, was converted to minutes by dividing by 60. The trip distance was converted to kilometers from miles by multiplying it by 1.60934. When calculating speed in km per hour, trip duration was divided by 3600 to be converted to hours.

Table 5: Answers to Business Question 2

		Yellow	Green
Trip duration	Average	15.06	14.24
	Median	11.87	11.08
	Minimum	3	3

	Maximum	120	120
Trip distance	Average	5.1	4.98
	Median	2.9	3.27
	Minimum	0.8	0.8
	Maximum	80.47	80.47
Speed	Average	18.75	20.23
	Median	16.42	18.31
	Minimum	0.4	0.4
	Maximum	88.51	88.51

avg_trip_duration	median_trip_duration	min_trip_duration	max_trip_duration	avg_distance_km	median_distance_km	min_distance_km	max_distance_km	avg_speed_kmph	median_speed_kmph	min_speed_kmph	max_speed_kmph
15.06	11.87	3	120	5.1	2.9	0.8	80.47	18.75	16.42	0.4	88.51

avg_trip_duration	median_trip_duration	min_trip_duration	max_trip_duration	avg_distance_km	median_distance_km	min_distance_km	max_distance_km	avg_speed_kmph	median_speed_kmph	min_speed_kmph	max_speed_kmph
14.24	11.08	3	120	4.98	3.27	0.8	80.47	20.23	18.31	0.4	88.51

Figure 5.2: Screenshots of business question 2 result

Question 3:

For each taxi color (yellow and green), each pair of pickup and dropoff boroughs, each month, each day of the week, and each hour:

(a) What was the total number of trips?
(b) What was the average distance?
(c) What was the average amount paid per trip?
(d) What was the total amount paid?

PUBorough	DOBorough	year_month	day_of_week	hour_of_day	total_trips	avg_distance	avg_amount_per_trip	total_amount_paid
Bronx	Bronx	2015-01	Friday	0	39	3.32	14.33	558.94
Bronx	Bronx	2015-01	Friday	1	28	3.09	13.71	384.01
Bronx	Bronx	2015-01	Friday	2	29	2.91	12.95	375.64
Bronx	Bronx	2015-01	Friday	3	26	2.4	11.82	307.38
Bronx	Bronx	2015-01	Friday	4	29	2.77	12.45	360.92
Bronx	Bronx	2015-01	Friday	5	18	2.89	12.68	228.3
Bronx	Bronx	2015-01	Friday	6	32	2.33	10.4	332.85
Bronx	Bronx	2015-01	Friday	7	51	2.16	11.92	608.13
Bronx	Bronx	2015-01	Friday	8	55	2.08	12.01	660.32
Bronx	Bronx	2015-01	Friday	9	20	2.28	12.99	259.81

Figure 5.3: Screenshot of business question 3 (yellow taxi)

PUBorough	DOBorough	year_month	day_of_week	hour_of_day	total_trips	avg_distance	avg_amount_per_trip	total_amount_paid
Bronx	Bronx	2015-01	Friday	0	290	2.36	9.81	2846.11
Bronx	Bronx	2015-01	Friday	1	213	2.13	9.77	2080.37
Bronx	Bronx	2015-01	Friday	2	142	2.24	9.7	1376.96
Bronx	Bronx	2015-01	Friday	3	88	3.48	11.81	1039.18
Bronx	Bronx	2015-01	Friday	4	64	2.57	9.75	624.28
Bronx	Bronx	2015-01	Friday	5	68	2.77	9.75	663.1
Bronx	Bronx	2015-01	Friday	6	160	2.8	10.29	1646.16
Bronx	Bronx	2015-01	Friday	7	532	2.1	10.46	5566.12
Bronx	Bronx	2015-01	Friday	8	841	2.29	11.39	9578.48
Bronx	Bronx	2015-01	Friday	9	590	2.24	10.61	6258.57

Figure 5.4: Screenshot of business question 3 (green taxi)

Question 4: What was the percentage of trips where drivers received tips?

---The percentage of trips where drivers received tips is 64.4%---

1.2 percentage_trips_with_tips
64.41972580598217

Figure 5.5: Business question 4

Question 5: For trips where drivers received tips, what percentage had tips of at least $5?

---For trips where the driver received tips, 8.2% tips were at least $5---

1.2 percentage_trips_with_tips_at_least_5_dollars
8.210529168251588

Figure 5.6: Business question 5

Question 6: Average speed and distance analysis

Trips were classified into the following duration bins:

- Under 5 minutes
- From 5 to 10 minutes
- From 10 to 20 minutes
- From 20 to 30 minutes
- From 30 to 60 minutes
- At least 60 minutes

For each duration bin:

(a) Calculate the average speed (in kilometers per hour).
(b) Calculate the average distance per dollar.

Generally, this analysis ultimately seeks to maximize profitability by understanding which types of trips are most lucrative for taxi drivers. The average speed and average distance per dollar of each duration bins are shown in the last 2 columns (figure 5.7)

ABC duration_bin	1.2 percentage_trips_with_tips	1.2 percentage_trips_with_tips_at_least_5_dollars	1.2 avg_speed_kmph	1.2 avg_distance_per_dollar_kmpd
From 5 mins to 10 mins	62.9163	0.4501	17.21	0.21
Under 5 Mins	59.2211	0.3822	19.69	0.18
From 10 mins to 20 mins	65.9989	3.9366	17.71	0.26
From 20 mins to 30 mins	66.6003	29.584	21.16	0.31
At least 60 mins	58.9663	94.8876	22.74	0.5
From 30 mins to 60 mins	66.052	76.348	25.62	0.37

Figure 5.7: Business question 6

Question 7:

Based on the analysis of duration bins, which bin would be most advisable for a taxi driver to target to maximize income?

---

duration_bin	total_income	total_trip_hours	avg_income_per_hours
From 5 mins to 10 mins	2017173348.9330919	24945013.06611111	80.86479424111867
Under 5 Mins	470264353.0107109	4219329.1605555555	111.45476807237091
From 10 mins to 20 mins	3788833500.4585156	57032258.425	66.43316616053357
From 20 mins to 30 mins	2316268478.0617347	35108884.25861111	65.97385610434563
At least 60 mins	383852159.52006483	6903671.509722223	55.60116221918988
From 30 mins to 60 mins	2326355309.49347	33539491.726944443	69.36167454274683

Figure 5.8: Business question 7

Based on the query results from figure 5.8, although trips within the "Under 5 mins" duration bin generate less total income overall, they yield the highest earnings per hour, which is key for a driver aiming to maximize efficiency. Therefore, a taxi driver should target the "Under 5 mins" trip.

SECTION 6.
Machine Learning Model Development

6.1 Problem statement

In this stage of the project, I aim to develop machine learning models that can accurately predict the total fare amount for taxi trips in New York City based on various factors derived from previous steps. The goal of this stage is to provide a robust predictive model which can help stakeholders in understanding fare dynamics and optimizing pricing strategies.

To achieve this, we will follow a structured approach:

1. **Baseline model development**: build a baseline model by using the average fare amount calculated previously. I will compute the Root Mean Square Error (RMSE) for this baseline model to quantify its predictive accuracy.
2. **Model selection and training:** Two machine learning models, including Linear Regression Random Forest will be trained using Spark ML pipelines. I will use all data except for trips occurring in October, November, and December 2022 for training and validation purposes, and use RMSE score to evaluate the models.
3. **Model evaluation and comparison:** In this stage, performances of multiple machine learning models are compared based on criteria such as processing time, complexity, accuracy, etc. to identify the best model.
4. **Final prediction and performance assessment**

6.2 Baseline model

```
# Join with the train DataFrame to add the baseline prediction to each trip
df_with_baseline_train = train_validation.join(avg_amount, on='taxi_color', how='left')

# Baseline prediction: use the average amount per trip for each taxi color in the training DataFrame
df_with_baseline_train = df_with_baseline_train.withColumn('baseline_prediction', col
('avg_amount_per_trip'))

df_with_baseline_validation = test.join(avg_amount, on='taxi_color', how='left')

# Baseline prediction
df_with_baseline_validation = df_with_baseline_validation.withColumn('baseline_prediction', col
('avg_amount_per_trip'))

# Calculate RMSE for the baseline model on the validation set
baseline_evaluator = RegressionEvaluator(labelCol='total_amount', predictionCol='baseline_prediction',
metricName='rmse')
baseline_rmse = baseline_evaluator.evaluate(df_with_baseline_validation)
print(f"Baseline Model RMSE: {baseline_rmse}")
```

```
▸ (4) Spark Jobs
▸ df_with_baseline_train: pyspark.sql.dataframe.DataFrame
▸ df_with_baseline_validation: pyspark.sql.dataframe.DataFrame
Baseline Model RMSE: 18.79564997547481
```

Figure 6.1: Baseline model

6.3 Feature selection

The target feature is total_amount.

Selected features are: 'passenger_count', 'trip_distance', 'extra', 'mta_tax', 'tip_amount', 'improvement_surcharge', 'congestion_surcharge', 'airport_fee', 'trip_duration', 'speed'

6.4 Data processing

Data processing is a crucial step in the machine learning workflow, particularly before modelling. The quality and integrity of the data significantly impact the performance of any predictive model. The following steps were undertaken to pre-process the data for our machine learning models:

- Data type conversion: Some columns are converted to ensure the correct data types for analysis and modelling. Specifically, I used DoubleType for trip_duration and IntegerType for passenger_count.
- Null values handling: I employed the Imputer, which handles missing values by replacing them with mean of their respective columns to maintain data integrity while preventing information loss.

6.5 Modelling

The train_validation set is used for modelling, involves all data except the last 3 months of 2022 (October, November, and December), while the test set used data of those 3 months. train_validation set was splitted, with 80% for training and 20% for validating. The models have been fitted on training data. Validation data was used for validating, and test data has been used for making predictions of the models. Linear Regression and Random Forest are evaluated using RMSE score.

```
# Linear Regression
linear = LinearRegression(featuresCol="features", labelCol=target_feature)

# Fitting model on the train data
lreg_model = linear.fit(train)
```

Figure 6.2: Linear Regression model

```
# Random Forest
randomforest = RandomForestRegressor(featuresCol="features", labelCol=target_feature)

# Fitting model on the train data
rf_model = randomforest.fit(train)
```

Figure 6.3: Random Forest model

6.6 Results and evaluation

Table 6: Models comparison

	Baseline model	Linear Regression	Random Forest
RMSE	18.7956	30.5324	5.5627
Processing time	5 minutes	20 minutes	1 hour 10 minutes
Complexity	Simple	Not suitable for complex data	More complex, suitable for complex data

Despite taking more time to process, Random Forest model performed the best with an RMSE of 5.56, significantly lower than both the baseline (18.73) and Linear Regression (30.53)

SECTION 8.
Conclusion

In this project, I utilised various advanced technologies, including Microsoft Azure, Databricks, Apache Spark, PySpark, and Spark SQL, to process and analyze large datasets of yellow and green taxi trips from 2015 to 2022. The integration of these tools allowed us to perform data ingestion, transformation, and analysis efficiently. By employing PySpark, dataset was cleaned and transformed to ensure data quality and reliability. The use of Spark SQL enabled data analyst to perform complex queries and derive meaningful insights, which are critical for understanding taxi trip patterns and trends in New York City.

Additionally, the project builds predictive models, including Linear Regression and Random Forest which may enhance decision-making processes within the taxi service industry. The findings contribute to the broader understanding of urban mobility and can assist stakeholders in optimizing operations, improving service efficiency, and maximizing profitability.

REFERENCES

City of New York. (n.d.). Official website of the City of New York. https://www.nyc.gov/

Sixt. (n.d.). *Driving in New York State*. Sixt Magazine. https://www.sixt.com/magazine/tips/driving-in-new-york-state/

APPENDICES

```
%sql
SELECT
  DATE_FORMAT(tpep_pickup_datetime, 'yyyy-MM') AS year_month,
  COUNT(*) AS total_trips,
  DATE_FORMAT(tpep_pickup_datetime, 'EEEE') AS day_of_week,
  HOUR(tpep_pickup_datetime) AS hour_of_day,
  AVG(passenger_count) AS avg_passenger,
  AVG(total_amount) AS avg_total_amount,
  AVG(total_amount / passenger_count) AS avg_amount_per_passenger
FROM data_table
GROUP BY year_month, day_of_week, hour_of_day
ORDER BY year_month, total_trips DESC
```

Figure SQL queries for business question 1

```
%sql
SELECT
    ROUND(AVG(trip_duration/60), 2) AS avg_trip_duration,
    ROUND(PERCENTILE(trip_duration/60,0.5),2) AS median_trip_duration,
    ROUND(MIN(trip_duration/60), 2) AS min_trip_duration,
    ROUND(MAX(trip_duration/60), 2) AS max_trip_duration,
    ROUND(AVG(trip_distance*1.60934), 2) AS avg_distance_km,
    ROUND(PERCENTILE(trip_distance*1.60934, 0.5), 2) AS median_distance_km,
    ROUND(MIN(trip_distance*1.60934), 2) AS min_distance_km,
    ROUND(MAX(trip_distance*1.60934), 2) AS max_distance_km,
    ROUND(AVG((trip_distance*1.60934) / (trip_duration/3600)), 2) AS avg_speed_kmph,
    ROUND(PERCENTILE((trip_distance*1.60934) / (trip_duration/3600), 0.5), 2) AS median_speed_kmph,
    ROUND(MIN((trip_distance*1.60934) / (trip_duration/3600)), 2) AS min_speed_kmph,
    ROUND(MAX((trip_distance*1.60934) / (trip_duration/3600)), 2) AS max_speed_kmph
FROM yellow_table
```

Figure SQL queries for business question 2 - Yellow

```
%sql
SELECT
    ROUND(AVG(trip_duration/60), 2) AS avg_trip_duration,
    ROUND(PERCENTILE(trip_duration/60,0.5),2) AS median_trip_duration,
    ROUND(MIN(trip_duration/60), 2) AS min_trip_duration,
    ROUND(MAX(trip_duration/60), 2) AS max_trip_duration,
    ROUND(AVG(trip_distance*1.60934), 2) AS avg_distance_km,
    ROUND(PERCENTILE(trip_distance*1.60934, 0.5), 2) AS median_distance_km,
    ROUND(MIN(trip_distance*1.60934), 2) AS min_distance_km,
    ROUND(MAX(trip_distance*1.60934), 2) AS max_distance_km,
    ROUND(AVG((trip_distance*1.60934) / (trip_duration/3600)), 2) AS avg_speed_kmph,
    ROUND(PERCENTILE((trip_distance*1.60934) / (trip_duration/3600), 0.5), 2) AS median_speed_kmph,
    ROUND(MIN((trip_distance*1.60934) / (trip_duration/3600)), 2) AS min_speed_kmph,
    ROUND(MAX((trip_distance*1.60934) / (trip_duration/3600)), 2) AS max_speed_kmph
FROM green_table
```

SQL queries for business question 2 – Green

```
%sql
SELECT
    PUBorough,
    DOBorough,
    DATE_FORMAT(tpep_pickup_datetime, 'yyyy-MM') AS year_month,
    DATE_FORMAT(tpep_pickup_datetime, 'EEEE') AS day_of_week,
    HOUR(tpep_pickup_datetime) AS hour_of_day,
    COUNT(*) AS total_trips,
    ROUND(AVG(trip_distance), 2) AS avg_distance,
    ROUND(AVG(total_amount), 2) AS avg_amount_per_trip,
    ROUND(SUM(total_amount), 2) AS total_amount_paid
FROM green_table
GROUP BY PUBorough, DOBorough, year_month, day_of_week, hour_of_day
ORDER BY PUBorough, DOBorough, year_month, day_of_week, hour_of_day
```

SQL queries for business question 3 – Green

```
%sql
SELECT
    PUBorough,
    DOBorough,
    DATE_FORMAT(tpep_pickup_datetime, 'yyyy-MM') AS year_month,
    DATE_FORMAT(tpep_pickup_datetime, 'EEEE') AS day_of_week,
    HOUR(tpep_pickup_datetime) AS hour_of_day,
    COUNT(*) AS total_trips,
    ROUND(AVG(trip_distance), 2) AS avg_distance,
    ROUND(AVG(total_amount), 2) AS avg_amount_per_trip,
    ROUND(SUM(total_amount), 2) AS total_amount_paid
FROM yellow_table
GROUP BY PUBorough, DOBorough, year_month, day_of_week, hour_of_day
ORDER BY PUBorough, DOBorough, year_month, day_of_week, hour_of_day
```

SQL queries for business question 3 – Yellow

```sql
%sql
SELECT (COUNT(CASE WHEN tip_amount > 0 THEN 1 END) * 100 / COUNT(*)) AS percentage_trips_with_tips
FROM data_table;
```

SQL queries for business question 4

```sql
%sql
SELECT (COUNT(CASE WHEN tip_amount >= 5 THEN 1 END) * 100 / COUNT(*)) AS
percentage_trips_with_tips_at_least_5_dollars
FROM data_table;
```

SQL queries for business question 5

```sql
%sql
WITH trip_statistics AS (
    SELECT
        CASE
            WHEN trip_duration < 300 THEN 'Under 5 Mins'
            WHEN trip_duration >= 300 AND trip_duration < 600 THEN 'From 5 mins to 10 mins'
            WHEN trip_duration >= 600 AND trip_duration < 1200 THEN 'From 10 mins to 20 mins'
            WHEN trip_duration >= 1200 AND trip_duration < 1800 THEN 'From 20 mins to 30 mins'
            WHEN trip_duration >= 1800 AND trip_duration < 3600 THEN 'From 30 mins to 60 mins'
            ELSE 'At least 60 mins'
        END AS duration_bin,
        (trip_distance * 1.60934) / (trip_duration / 3600) AS speed,
        (trip_distance * 1.60934) / total_amount AS distance_per_dollar,
        tip_amount
    FROM data_table
    WHERE trip_duration > 0
)

SELECT
    duration_bin,
    ROUND((SUM(CASE WHEN tip_amount > 0 THEN 1 ELSE 0 END) * 100 / COUNT(*)), 4) AS percentage_trips_with_tips,
    ROUND((SUM(CASE WHEN tip_amount >= 5 THEN 1 ELSE 0 END) * 100 / SUM(CASE WHEN tip_amount > 0 THEN 1 ELSE 0
    END)), 4) AS percentage_trips_with_tips_at_least_5_dollars,
    ROUND(AVG(speed), 2) AS avg_speed_kmph,
    ROUND(AVG(distance_per_dollar), 2) AS avg_distance_per_dollar_kmpd
FROM trip_statistics
GROUP BY duration_bin;
```

SQL queries for business question 6

```sql
%sql
WITH trip_statistics AS (
    SELECT
        CASE
            WHEN trip_duration < 300 THEN 'Under 5 Mins'
            WHEN trip_duration >= 300 AND trip_duration < 600 THEN 'From 5 mins to 10 mins'
            WHEN trip_duration >= 600 AND trip_duration < 1200 THEN 'From 10 mins to 20 mins'
            WHEN trip_duration >= 1200 AND trip_duration < 1800 THEN 'From 20 mins to 30 mins'
            WHEN trip_duration >= 1800 AND trip_duration < 3600 THEN 'From 30 mins to 60 mins'
            ELSE 'At least 60 mins'
        END AS duration_bin,
        (trip_distance * 1.60934) / (trip_duration / 3600) AS speed,
        (trip_distance * 1.60934) / total_amount AS distance_per_dollar,
        tip_amount,
        total_amount, trip_duration
    FROM data_table
    WHERE trip_duration > 0
)
SELECT
    duration_bin,
    SUM(total_amount) AS total_income,
    SUM(trip_duration)/3600 AS total_trip_hours,
    SUM(total_amount) / (SUM(trip_duration)/3600) AS avg_income_per_hours
FROM trip_statistics
GROUP BY duration_bin
```

SQL queries for business question 7