Big Data Engineering: Assignment 2

Big Data Processing with Databricks Spark.

Oliver Salman 13881750

Table of Contents

Project Overview

This project has an aim to uncover the dynamics of how customers ride taxis in NYC, and as a result, this project will develop methods on how taxi prices can be calculated and predicted. This project has an emphasis on creating solutions that are both scalable and efficient, which is why we are using Databricks alongside Apache Spark. Both Spark and Databricks employ a distributed framework, meaning commands are distributed across multiple clusters to handle massive volumes of data. Throughout this project, we will discuss some of the techniques used to allow for an efficient and scalable system, as well as architectural implementations and possible improvements for future iterations for this project.

Part 1: Data Ingestion and Preparation

The first stage of this project includes downloading all necessary data from the TLC website to our local machine and uploading it to a Microsoft Azure Container. An Azure Container is an efficient way of storing huge amounts of data, with integration capabilities with distributed frameworks like DataBricks.

[1.1] Data Format

We have decided to convert the TLC datasets from csv to parquet format. Parquet is a column based file format that consists of row groups, header, and footer, and in each row group data in the same columns are stored together. Parquet files are compressed to a significantly smaller size which makes them faster to read and cheaper to store. Generally, parquet files are one order of magnitude smaller than their equivalent csv. Parquet files also have an inferred schema, meaning the data types are clearly defined and will not get lost in human conversion. CSV files do not have an inferred schema, meaning each column data type has to be defined manually, which can lead to data type mismatches. Overall, our project working with big data makes it clear that utilizing parquet files over csv files has many advantages.

[1.2] Cleaning

An important stage in our project was to properly transform our data into a clean, usable format. Transformation of our raw data not only increases the quality of our data, but also reduces the total size of our data, making analytics more efficient. We first started cleaning our data by producing summary statistics of all the columns in the datasets. This is where we can filter out illogical data (etc. trip distance being less than 0), extreme values (e.g trip distance being 1000+ miles) and null values. After the data has been cleansed, we then move onto merging the schemas of both datasets (for yellow and green taxi types). This required us to firstly create an empty table that includes all columns that are in both datasets, standardizing column names, and finally inserting our data into this new table if conditions have been met. This added complexity of merging schemas is a downside of using the parquet file format, as schema modifications and evolutions are not supported. Future iterations of this project can consider using AVRO file format for its capabilities to handle evolving schemas.

[1.3] Features

To streamline the analytics of the project, features were added to our dataset that will help with more efficient and error prone queries. The features added were inferred from other columns that were provided. The features added to our dataset include year, filename, taxi type, the difference in seconds between dropoff time and pickup time, as well as speed in mph.

Part 2: Business Questions

The following questions in this section revolve around business related data analysis. Each question will be listed and its corresponding answer will be discussed with supporting claims.

[2.1] Question 1

For each year and month:

- **What was the total number of trips?** In the results, 2019 consistently had the highest total number of trips across all months versus 2020 and 2021. A plausible reason for this is the spike of covid-19 in 2020 and 2021 with less people catching taxis and public transport in general.

	year	month	tot_trips
1	2019	1	8032786
2	2019	2	5125469
3	2019	3	5554256
4	2019	4	5228068
5	2019	5	5347092
6	2019	6	4935304
7	2019	7	4505296

- **Which day of week (e.g. monday, tuesday, etc..) had the most trips?** The days of the week which had most trips were well distributed, showing no obvious trends.

	year	month	day	count	rk
1	2019	11	7	855639	1
2	2019	2	6	880900	1
3	2019	3	6	969371	1
4	2019	4	3	878006	1
5	2019	5	5	926548	1
6	2019	6	7	827437	1
7	2019	7	4	787674	1

- **Which hour of the day had the most trips?** On the other hand, the hours of the day that had most trips had a clear trend, with 6pm having the most trips. This was an exception during April 2020 and April 2021 where 3pm was the most popular time. A reason for this obvious change is the mayor of New York City in March 2020 declared NYC would be temporarily shut down due to the exponential rise of covid cases. In April 2021, the mayor announced NYC would be fully reopened by July 1.

	year	month	hour	count	rk
1	2020	3	18	149905	1
2	2021	6	18	137374	1
3	2019	2	18	340643	1
4	2019	3	18	363937	1
5	2019	1	18	536176	1
6	2019	5	18	350655	1
7	2019	6	18	308012	1

- **What was the average number of passengers?** The average number of passengers for each year and month varied between 1.5 to 1.7 passengers. 2019 had the highest averages being around 1.7. 2020 and 2021 had the lowest averages during peak times of covid cases in April 2020 and 2021.

	year	month	average
1	2019	1	1.5715857487053682
2	2019	2	1.7128290113548634
3	2019	3	1.7236074462538278
4	2019	4	1.7184646412403206
5	2019	5	1.7125725534552239
6	2019	6	1.6999066318913687
7	2019	7	1.6995369449643265

- **What was the average amount paid per trip?** The average price paid per trip was fairly stable across the 3 years with averages varying between $15 and $19. There were no apparent trends in the price fluctuations.

	year	month	average
1	2019	1	15.303587
2	2019	2	18.396180
3	2019	3	18.995435
4	2019	4	18.988574
5	2019	5	19.343212
6	2019	6	19.477962
7	2019	7	19.219752

- **What was the average amount paid per passenger?** As expected, the average amount paid per passenger was fairly stable, varying between $10 and $12. There were no apparent trends in price paid per passenger.

	year	month	average
1	2019	1	9.737672292209764
2	2019	2	10.740231440526836
3	2019	3	11.020743175185046
4	2019	4	11.049732152937862
5	2019	5	11.294827749617873
6	2019	6	11.45825402088597
7	2019	7	11.308816826222877

[2.2] Question 2

For each taxi color (yellow and green):

- **What was the average, median, minimum and maximum trip duration in seconds?** The difference in time was calculated in part 1 and was added as a feature to the dataset to remove any inconsistencies in the business questions. Constraints were added to exclude illogical and extreme values, trip ride times are positive and less than 10800 seconds (3 hours). The average for yellow taxis is 835 seconds 838 seconds for green taxis (roughly 15 minute), with the median of both being roughly 10 minutes. Below is the output of this query.

	taxi_type	AVG	MED	MIN	MAX
1	yellow	835.3005122166585	655	1	10800
2	green	868.355557621668	656	1	10792

- **What was the average, median, minimum and maximum trip distance in km?** Similarly, the trip distance in km was calculated by multiplying the given trip distance (in miles) to kilometers by approximately 1.6. The average, median, minimum and maximum of the distance in kilometers produced an output as follows.

	taxi_type	AVG	MED	MIN	MAX
1	yellow	3.019712165	1.68	0	120.70080000000
2	green	3.094818838	1.86	0	120.05706240000

- **What was the average, median, minimum and maximum speed in km per hour?** The speed (mph) was calculated in part 1 and was added as a feature to the dataset to remove any inconsistencies in the business questions. The speed had constraints applied so values must be between 0 and 100 mph (non inclusive). The speed in mph

was then converted to kph by multiplying by approximately 1.6. The average, median, minimum and maximum speed in kmph produced an output as follows.

	taxi_type	AVG	MED	MIN	MAX
1	green	19.44	10.78	0.01	160.71
2	yellow	19.14	10.34	0.01	160.84

[2.3] Question 3 and 4

- **What was the percentage of trips where the driver received tips?** This question was answered by counting the cases when a passenger tipped their driver, divided by the total number of trips. The result of this query gave us that approximately 67% of passengers tipped the driver.

	percentage_tipped
1	67.63

- **For trips where the driver received tips, what was the percentage where the driver received tips of at least $10.** This question was answered similarly to the first question, firstly counting the cases when a passenger tipped the driver over $10, divided by the cases when a passenger tipped the driver at all. The result of this query gave us that approximately 3.4% of passengers tipped the driver at least $10.

	percentage_tipped_over_10
1	3.4

[2.4] Question 5

Classify each trip into bins of durations: Under 5 Mins, from 5 mins to 10 mins, from 10 mins to 20 mins, from 20 mins to 30 mins, at least 30 mins

- This was achieved by finding the cases where the trip distance satisfied the above conditions. A count over all instances of the cases shows that the trips are correctly partitioned into their respective bins. The following output shows the count of taxi trips in each bin.

	bucket	count
1	At least 30 mins	7068858
2	From 10 mins to 20 mins	31218968
3	From 20 mins to 30 mins	10893227
4	From 5 mins to 10 mins	27217455
5	Under 5 Mins	13387787

For each of these bins, calculate:

- **Average speed (km per hour).** Mentioned previously, the speed was calculated in part 1 as an inferred variable from trip distance and time in hours. Multiplying by approximately

1.6 gave the average speed in kmph. The output shows that speed doesn't have a noticeable correlation between any of the bins.

- **Average distance per dollar (km per $).** The average distance per dollar was calculated by dividing the average distance traveled by the average total amount. These results show some correlation between average distance per dollar and the bins. For trips in the 'under 5 mins' bin show a low average distance per dollar which can be explained by the possibility of heavy traffic in highly populated areas like Manhattan. This is compared to trips in the 'at least 30 mins' bin with a high average distance per dollar, as most of these rides would have a high trip distance, meaning highways and motorways could have been used.

	bucket	avg_speed	avg_distance_per_dollar
1	At least 30 mins	27.13	0.37
2	From 10 mins to 20 mins	17.83	0.24
3	From 20 mins to 30 mins	22.17	0.31
4	From 5 mins to 10 mins	17.07	0.18
5	Under 5 Mins	19.86	0.12

[2.5] Question 6

Which duration bin will you advise a taxi driver to target to maximize his income? To maximize income, the taxi driver must strive to minimize the average distance per dollar and trip duration. This can be achieved by residing in congested areas such as city metros where distance between pickup location and dropoff locations are relatively close. This will increase the frequency of new passengers, maximizing total trips. Additionally, taking passengers in congested areas such as city metros minimizes the average distance per dollar passengers pay as heavy traffic will reduce the average speed of a taxi, making city trips generally more expensive for passengers, hence maximizing income for taxi drivers.

Part 3: Machine Learning

This section of the report will showcase two machine learning models that will predict a fair taxi price for taxi rides in 2021. This report will go into depth around the major components that these models use, scaling and efficiency techniques, as well as evaluation metrics and a final discussion.

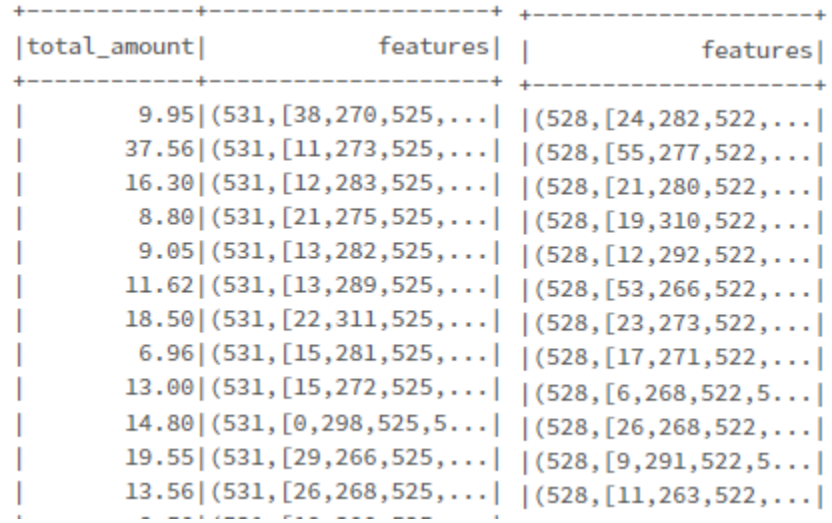
Data Cleaning and Preparation.

Data cleaning and preparation for machine learning differs from the cleaning and preparation discussed earlier. We start by converting our pickup and dropoff times from a date time data type to a unix standard date. This conversion from date time to unix time will be much easier to parse to a transformer, making this an efficient technique working with date-time objects. We

also dropped some features that will not contribute to our prediction. Though, in practice, feature selection methods such as backward selection should be used to choose what features to include. Due to environment and time restrictions, features were chosen to be removed in the cleaning phase. Then, categorical variables are 'One Hot Encoded' using a string indexer. One Hot Encoding (OHE) refers to translating a categorical variable into a series of numerical bins.

Pipeline

The Spark Pipeline defines a set of instructions on how large volumes of data are represented for the machine learning model. We define a vector assembler that takes in our OHE categorical variables (mentioned above) as well as a list of our numerical variables, and creates a vector describing these. The vector assembler is a method to reduce the dimensionality of our model by condensing our features into a single column. From here, we add our vectorized features and our label to a stage and parse it to a pipeline. Both training and testing dataset were then transformed into the vectorised format defined by the pipeline. The outputs of the transformed datasets are shown below.



total_amount	features	features
9.95	(531,[38,270,525,...	(528,[24,282,522,...
37.56	(531,[11,273,525,...	(528,[55,277,522,...
16.30	(531,[12,283,525,...	(528,[21,280,522,...
8.80	(531,[21,275,525,...	(528,[19,310,522,...
9.05	(531,[13,282,525,...	(528,[12,292,522,...
11.62	(531,[13,289,525,...	(528,[53,266,522,...
18.50	(531,[22,311,525,...	(528,[23,273,522,...
6.96	(531,[15,281,525,...	(528,[17,271,522,...
13.00	(531,[15,272,525,...	(528,[6,268,522,5...
14.80	(531,[0,298,525,5...	(528,[26,268,522,...
19.55	(531,[29,266,525,...	(528,[9,291,522,5...
13.56	(531,[26,268,525,...	(528,[11,263,522,...

Model 1: Linear Regression

The dataset is then split into two partitions, one partition being the 2019 and 2020 data (for training) and the 2021 data (for predictions). We then fit and transform 2019/2020 data to our pipeline. We then define our linear regression model using Sparks Linear Regression module, and declare the features column and label column. We then fit the linear regression model with the training datasets' features. Then we were able to use the trained Linear Regression to make predictions on the training set, and then finally, make predictions on the testing set. Below is an output of our test predictions.

```
+------------+--------------------+------------------+
|total_amount|            features|        prediction|
+------------+--------------------+------------------+
|        2.80|(531,[4,308,525,5...|3.3661146511262814|
|        2.85|(531,[29,283,525,...|10.349661111104524|
|        2.95|(531,[37,326,525,...|15.319190746399727|
|        3.30|(531,[0,262,525,5...|  6.22192280408035|
|        3.30|(531,[0,262,525,5...| 6.633860625305715|
|        3.30|(531,[0,262,525,5...| 6.409084375035945|
|        3.30|(531,[0,262,525,5...|6.1901523281430855|
|        3.30|(531,[0,262,525,5...| 6.395163422753285|
|        3.30|(531,[0,263,525,5...| 6.419210051624759|
|        3.30|(531,[0,263,525,5...| 6.738475908742657|
+------------+--------------------+------------------+
```

Model Evaluation

Evaluating the model consisted of creating a Regression Evaluator using the prediction column and label columns, and specifying the evaluation metric, which will be Root Mean Squared Error metric (RMSE). The RMSE evaluation metric calculates the average distance between the predicted values from the model and the actual values in the dataset. The lower the RMSE score, the better the model is at fitting a dataset. Firstly, the RMSE metric was calculated for the models training set, and then testing set. The RMSE for the training set was 3.25 and 1574.31 for the test set.

Any issues you faced and how you solved them

The first issue faced was not having consistent data in part 1 when cleaning the parquet files. This issue arose as a result of my lack of knowledge of the parquet data format, and I would often get resulting parquet files that were either empty, did not have the correct number of rows. This was overcome by firstly researching and understanding how parquet files are manipulated in large scale data analytics. This was implemented in my code by clearly defining the stages in the cleaning process and creating ‘iterations’ of my data by writing and then reading the parquet files.

Another issue that I faced was not being able to fit my machine learning model to the regressor. After research about databricks and the clusters, I found out that the size and dimensionality of the dataset was too large for the community version of databricks. I overcame this issue by reducing the number of features and employing a simpler model. Additionally, another issue identified was the exceedingly large RMSE score for the testing data. I found for the majority of the calculations to be fairly accurate, except for the minimum prediction of -$6M, which as a result, also malformed the std deviation. Further investigation into the exceedingly large minimum value must be carried out.

summary	total_amount	prediction
count	15950447	15950447
mean	18.455386	18.060296223163558
stddev	14.125647881896606	1574.3580609774983
min	2.60	-6287417.822968636
max	949.80	866.0271283024922