Big Data Engineering

Master of Data Science and Innovation

University Technology of Sydney

Report

Data processing on a big dataset with Spark

Javiera de la Carrera Garcia
13743354

Objective

The main objective of this report is to analyse two years of the TLC trips dataset using Spark. For that, we had to load the available data, perform data transformation and analysis on it, and finally train a Machine Learning model to predict the total amount of a trip.

Assumptions and constraints

Because the dataset given has a high-level description of the variables, some general assumptions were made to clean the data. First, all variables could not be negative. Second, because the large amount of data, it was assumed that the numeric variables had a Gaussian distribution. Other specific assumptions are described in the cleaning section below.

Technical Issues

Some technological constraints related to the size of the dataset were fixed. First, because the large amount of data, it was necessary to specify the schema of each type of dataset (yellow and green taxis) to have a faster reading of all the csv files. Related to that, the Spark session was built without a "BroadcastJoinThreshold" (-1), because using a broadcast, the creation of the pipeline in the modelling part got an early stop[1]. Similarly, the "shuffle.partitions" was changed to 1000 because it works better than the default (200) for this size of the data[2]. Besides, to avoid Spark read and process all the data when the SQL queries were executed, the Spark dataframe was changed to a temporary view. Finally, to improve the process of training the models and parameter tuning, the modelling part was performed in Google Cloud Platform (GCP).

Data Understanding and cleaning

Two years (2019-2020) of green and yellow taxis trips were downloaded from the NYC page. After exploring it, it was realised that both taxi data has almost the same variables. Only yellow taxi's files had two fewer variables; therefore, they were created for these files. After that, the two taxis data were merged, and a new variable was created to identify the taxi type. This first data contained roughly 117 million rows and 21 columns. These last will be described below.

Numeric Variables[3]

First, the description' measures of these variables were explored. It was seen that all of them had non-logical negative values. Hence, these values were dropped. Then, the variables were cleaned considering the data dictionary from the NYC page. For example, the fare amount always starts at 2.50 dollars and the congestion and extra charge have a maximum of 2.75 and 1, respectively. See Appendix 1 for the first descriptions of the variables and the fare information.

Because after this first cleaning, there still some abnormal maximum values(outliers), they were taken out, assuming that the numeric variables have a normal distribution. Therefore, for each variable, all values higher than three or four standard deviations of its mean were dropped[4]. It is important to mention that exceptions were made to some variables because they had different behaviours. For example, tip amount had a very low standard deviation because trips with cash payment always had 0

[1]The problem was with Docker, not with CGP.
[2] Other numbers as 1500 and 2000 were tried but the performance was worse.
[3]Numeric variables: 'passenger_count', 'trip_distance', 'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount', 'improvement_surcharge', 'total_amount, 'congestion_surcharge', 'ehail_fee'.
[4] See Brownlee, J. (2020b, August 18).

tips; therefore, the maximum was put in 100 dollars. Also, mta tax was changed to a binary value, 0 or 0.5[5], and it was assumed a maximum of 6 passengers per car.

In addition, the last cleaning was made assuming logical trips. Hence, trips with zero distance and a minimum fare or trips with positive distance and fare were kept. See the minimum fares in Appendix 1.

Categorical Variables[6]

After exploring the categories inside each categorical variable, it could be seen that some of them had values no defined in the data dictionary given. For example, VendorID = 4 was changed to category 2 and RateCodeID =99 to 1. For the few missing values of trip_type it was assumed the same. See Appendix 2 for details.

Also, because only with credit card payment it was possible to record tips, the tip amount for the others payment types was changed to zero. Finally, Pickup and Dropoff LocationID were analysed in the data engineering part.

Datetime variables

The pick-up and drop-off datetime variables had values outside the sample used in this report (2019-2020). In the data engineering part, these rows were cleaned.

Data Engineering

The first variable analysed was Location ID. Because we only knew that there were 263 types of Locations IDs, the taxi zone lookout table from the webpage was merged to the actual data to have a better understanding[7]. It was realised that each ID corresponds to a Zone in a Borough. Therefore, because this last had few categories, it was used to analyse the total amount in different locations[8]. It was seen that different places have different fares, and EWR (Airport) was the most expensive Borough. See Appendix 3. Thus, because of that, Borough was a good candidate for the modelling part. Also, because it has few categories, it helped to have a faster pre-processing (one-hot-encoding part).

Besides, new variables were created from the drop-off datetime variable. There were built the year, month, day of the week, quarter and the hour variables. From them, only the year variable had outliers. Therefore, only data from 2019 and 2020 was kept. Also, because the hour of the day variable had 24 values, five bins were created to achieve the day's time.

On the other hand, the duration variable (in minutes) was created as the difference between each trip's drop off and pick-up. After looking at its description' values, it could be seen that the maximum value was almost 44,000 minutes. See Appendix 3. Therefore, the same assumption of the standard deviation was assumed to drop the outliers[9]. Also, as before, a new variable with bins was created to differentiate short and long trips.

Besides, three more variables were created, the average paid per passenger, a categorical variable with bins for trip distance and the speed(km/hour). This last one was cleaned assumed a maximum value of 220.

[5]Department of Taxation and Finance. (2019, March 19).
[6] Categorical variables: 'VendorID','RatecodeID','store_and_fwd_flag', 'PULocationID', 'DOLocationID', 'payment_type','taxi_type', 'trip_type'.
[7] https://www1.nyc.gov/assets/tlc/downloads/pdf/trip_record_user_guide.pdf
[8] All trips with Unknown Borough were dropped.
[9] After that, the maximum continued high. However, because no information was given about the duration of a trip, the maximum was not changed.

It is worth mentioning that the analysis between the variables and the target was made in the business questions and the data preparation parts.

Data Quality

After cleaning, there were roughly 70 million trips, and only the ehail fee variable had missing values. Because of that, this variable was not used in the models.

Exported data

The format of the optimised file to export the combined two years of data was selected considering the functionality of this new file. In this report, the use case was to answer some business question using SQL queries.

As we learnt in the lecture and as McDonald, C. (2019) mentions, "Parquet gives the fastest read performance with Spark". That is very important for our use case because it allows to compute the analytics and queries at different times without a reading-time constraint[10]. Therefore, despite it is known that the writing part is slow with this format, it was chosen as the first option because I have had a good experience with Parquet in the past. Then, after I realised the writing part did not take a long time, I decided not to change it.

The geniality behind Parquet for our use case is it is a column-based format. Therefore, given that we have to query some columns of the table, Parquet is better than rows-based format. As Bhatia, R. (2021) mentions, "Parquet increases the query performance as less seek time is required to go to the required columns and less IO is required as it needs to read only the columns whose data are required".

Business Questions

Besides the data understanding above, more data exploration was made through some business questions. Some of them are highlighted below, and the results are in the Notebook.

First, it could be realised that almost all the months in 2020 have fewer trips than 2019. That could be possible because probably Covid restrictions affected the number of taxi trips. Besides, as expected, most of the trips were at 6 pm because it is the post office hour.

Also, in the two years of the report, Thursday was the weekday that had most of the trips, and the average number of passengers was 2. Similarly, on average, trips in 2020 were cheaper than in 2019; therefore, the paid per passenger too.

On the other hand, comparing yellow and green taxis, the average duration for yellow was less, but both were near 850 seconds (14 minutes). In contrast, the average speed and distance were very similar between taxi types. Interestingly, the median of the duration and the distance was less than the average in both taxi types. That could happen because trips with a duration or distance near the maximum are few; therefore, they affected the average of these variables more than the median. See below this result for the duration.

[10] For example, I could answer the business questions in two separate days using Parquet files.

taxi_type	avg_dur	min_dur	max_dur	median_dur
yellow	843.0866080577625	1.0	16197.0	679.0
green	854.211110441199	1.0	16191.000000000002	679.0

Figure 1: duration measures for yellow and green taxis

On the other side, looking at the bins of different durations, it could be seen that the minimum duration (less than 5 minutes) have, on average, a higher speed than the bins of 5-10 and 10-20 minutes. That could be possible because it is more likely to be in traffic congestion (less speed) in larger trips.

duration_bins	ave_speed	ave_dist_dollar
>30	25.703337447297155	0.35497007534492064
<5	19.62316593317579	0.12833775722739388
10-20	17.436047735429053	0.2285887630802899
20-30	21.347825394595272	0.2856627274076294
5-10	16.803196441768144	0.17656291442405736

Figure 2: average speed and distance per dollar for duration bins

Finally, looking at the table above, the driver should target the duration bin that is most profitable considering distance per dollar. Hence, the driver should target trips of less than 5 minutes because it receives 1 dollar in 0.13 km of the journey.

Modelling and evaluation measures

Preparation of the data

An exploratory analysis was made to select the variables to use in the models. First, some variables were dropped because they have missing values and many categories (details in Appendix 4).

Second, the correlation between the numeric variables and the target was performed to understand their relationship. It was realised that trip distance, duration, tip amount and tolls amount had a positive correlation with the total amount. See details in the Notebook.

On the other hand, for categorical variables, the analysis was to look at the average of the total amount between their categories. As mentioned before, if a variable has different fares between its categories, it is useful to predict the target. After looking that, ratecodeID has different total fares in its classes, been the JFK Airport (N2) the most expensive. Besides, trip type, taxi type and mta tax were also considered valuable. See Appendix 5 for details. In contrast, categories in dates variables did not show a high difference in the total fare.

Therefore, considering that, two selection of variables were evaluated. One with all the raw variables (without the bins). The other, considering the relevant variables described above and the results of the feature importance from the first models[11]. See the details of each set in Appendix 6.

Because we are using a large dataset, some techniques were necessary to achieve a fast pre-processing. Mandatory from Spark, categorical variables were transformed to one-hot-encoders, and two vectors

[11] Given that in the first models the distance and duration were the most important variables, their bins never were used in the models

were the final inputs of the model (features and label). Besides, a pipeline was used to pre-process the data. After that, the data was split, considering only the last three months of 2020 for the test set.

Other valuable techniques were to use first default (simple) models and categorical variables with a small number of categories.

Models

Two types of models from SparkML were considered to achieve the objective, linear regressions and a Random Forest model. The linear model was used because it was possible a linear relationship between the predictors and the target. Contrary, the random forest performs better when there is a more complex relationship between them. Because a grid search is more time consuming, the parameter tuning for the last was computed manually. See details in the Notebook.

Evaluation

Regarding the objective, our models were evaluated in the test set with RMSE metric. RMSE is useful to compare models; however, because alone is meaningless, it was compared with the description values of the target (Li, S. 2018). After looking at the mean and standard deviation of the total fare amount, it could be realised that the RMSE was a logical value.

The best performance was achieved with the Linear Regression model and the Random Forest after parameter tuning with the first set of variables. Both had an RMSE of approximately two dollars. This result could be explained because it was expected a linear relationship between the predictors and the target (it was seen in the correlation, in the preparation part).

With the second set of variables, Random Forest achieved a better performance than the linear regression, but less than with the first set of variables. Therefore, when we are in a scenario with a possible linear relationship and Big Data, linear regressions are a good start because it is simple and less time-consuming than other models. If it is not the case, an option is to use a Random Forest model only with the relevant variables because the performance might not change too much[12]. However, the parameter tuning takes time. Besides, it is essential to mention that the cleaning part is crucial to the modelling part because, without that, the relationship between the target and the predictors can be eroded.

Another insight was the importance of the features in the project's objective. Because both models achieve the same performance, the feature importance of Random Forest was used because it is easy to understand. It could be realised that the numeric variables were the most important, highlighting trip distance and duration. Also, some categorical variables were relevant, like RatecodeID and Borough. See below.

[12] The RMSE obtained with the Random Forest (with parameter tuning) and the second set of variables was 2.2 dollars.

	idx	name	score
5	60	trip_distance	0.417131
6	61	duration	0.259664
4	59	tip_amount	0.100575
3	58	tolls_amount	0.058646
7	62	speed	0.038316
17	9	RatecodeID_ohe_2	0.033874
43	35	Borough_DO_ohe_Manhattan	0.023240
39	31	Borough_PU_ohe_Queens	0.019982
16	8	RatecodeID_ohe_1	0.017043
8	0	payment_type_ohe_1	0.007588

Figure 3: Feature importance for the best Random Forest (set with all variables)

Finally, it is worth mentioning that in both models, the test set has less RMSE than the train set, probably because of its size. It is only 5% of the total data, and therefore it might have less noise (weird cases of trips) than the train set. Also, the data distribution of them could not be the same, probably because 2020 data was fewer than 2019. Therefore, the given split affected the sets' performance, and for a real case, we need to consider that and perhaps change the split.

References

Wrg, A. (2021, April 15). Pyspark – demand forecasting data science project - Towards Data Science. Medium. https://towardsdatascience.com/pyspark-demand-forecasting-data-science-project-dae14b5319cc

Taxi Fare - TLC. (n.d.). Nyc.Gov. https://www1.nyc.gov/site/tlc/passengers/taxi-fare.page#:%7E:text=%242.50%20initial%20charge.,Dutchess%2C%20Orange%20or%20Putnam%20Counties.

TLC Trip Record Data - TLC. (n.d.). Nyc.Gov. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

McDonald, C. (2019, February 12). Tips and Best Practices to Take Advantage of Spark 2.x. Dzone.Com. https://dzone.com/articles/tips-and-best-practices-to-take-advantage-of-spark

Tan, L. (2019, February 13). NYC Taxi Data Analysis Part 1: Clean and Transform Data in BigQuery. Medium. https://medium.com/@linniartan/nyc-taxi-data-analysis-part-1-clean-and-transform-data-in-bigquery-2cb1142c6b8b

Wrg, A. (2021b, April 15). Pyspark – demand forecasting data science project - Towards Data Science. Medium. https://towardsdatascience.com/pyspark-demand-forecasting-data-science-project-dae14b5319cc

9. Regression — Learning Apache Spark with Python documentation. (n.d.). Github. https://runawayhorse001.github.io/LearningApacheSpark/regression.html#id6

Li, S. (2018, June 27). Building A Linear Regression with PySpark and MLlib. Medium. https://towardsdatascience.com/building-a-linear-regression-with-pyspark-and-mllib-d065c3ba246a

Han, J. (2019, March 5). Looking Through the Taxi Meter - Analysis of the NYC Green Taxi data. Medium. https://medium.com/@jiaminhan/looking-through-the-taxi-meter-analysis-of-the-nyc-green-taxi-data-c1dbe5619afe

Lin, T. (2021, January 16). Feature Selection Using Feature Importance Score - Creating a PySpark Estimator. Timlrx. https://www.timlrx.com/blog/feature-selection-using-feature-importance-score-creating-a-pyspark-estimator

Peluritis, T. (2020, May 13). 4 simple tips to improve your Apache Spark job performance! Medium. https://medium.com/swlh/4-simple-tips-to-improve-your-spark-job-performance-fbf586ce8f7d

Subair, H. (2020, January 17). Data Engineering with Apache Spark (Part 1) - hasifsubair. Medium. https://medium.com/hasifsubair/data-engineering-with-apache-spark-d4684fb6cf0b

Department of Taxation and Finance. (2019, March 19). Information on the taxicab and hail vehicle trip tax Tax . Nyc. https://www.tax.ny.gov/bus/mctmt/taxi.htm#:%7E:text=The%20taxicab%20and%20hail%20vehicle%20trip%20tax%20in%20the%20Metropolitan,and%20Bronx%20counties)%3B%20or

Brownlee, J. (2020b, August 18). How to Remove Outliers for Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/how-to-use-statistics-to-identify-outliers-in-data/

Bhatia, R. (2021, February 25). Big data file formats | AVRO | Parquet | Optimised Row Columnar (ORC) | Clairvoyant Blog. Medium. https://blog.clairvoyantsoft.com/big-data-file-formats-3fb659903271

Appendices

Appendix 1

i. First description of the variables without cleaning

summary	passenger_count
count	114827251
mean	1.5277970209353875
stddev	1.1775814890423952
min	0
max	9

summary	trip_distance
count	116825619
mean	3.3396968832882794
stddev	209.06998305117406
min	-37264.53
max	350914.88

summary	total_amount
count	116825619
mean	18.895122402629656
stddev	221.17702763633338
min	-1871.8
max	1084772.1

summary	fare_amount
count	116825619
mean	13.311462728042809
stddev	194.4980192640717
min	-1856.0
max	998310.0

summary	mta_tax
count	116825619
mean	0.49565033650707385
stddev	46.259569466555284
min	-0.5
max	500000.5

summary	ehail_fee
count	109047872
mean	5.364616508106777E-8
stddev	3.234347268731717E-4
min	0.0
max	1.95

summary	extra
count	116825619
mean	1.0469919483353511
stddev	46.2759172156658
min	-60.0
max	500000.8

summary	tip_amount
count	116825619
mean	2.0947549758564086
stddev	13.375502229783272
min	-493.22
max	141492.02

summary	improvement_surcharge
count	116825617
mean	0.29677402840246264
stddev	0.037349296637125454
min	-0.3
max	1.0

summary	tolls_amount
count	116825619
mean	0.35989980738236227
stddev	1.7515866650268808
min	-70.0
max	3288.0

summary	congestion_surcharge
count	110481057
mean	2.0925866952015135
stddev	0.9382563169971059
min	-2.75
max	4.5

ii. Standard Metered Fare used for the first cleaning

The minimum fare of 6.5 was calculated considering the fares + extras + congestion + improvement.

Standard Metered Fare

- **$2.50** initial charge.
- Plus **50 cents** per 1/5 mile when traveling above 12mph or per 60 seconds in slow traffic or when the vehicle is stopped.
- Plus **50 cents** MTA State Surcharge for all trips that end in New York City or Nassau, Suffolk, Westchester, Rockland, Dutchess, Orange or Putnam Counties.
- Plus **30 cents** Improvement Surcharge.
- Plus **50 cents** overnight surcharge 8pm to 6am.
- Plus **$1.00** rush hour surcharge from 4pm to 8pm on weekdays, excluding holidays.
- Plus New York State Congestion Surcharge of **$2.50** (Yellow Taxi) or **$2.75** (Green Taxi and FHV) or **75 cents** (any shared ride) for all trips that begin, end or pass through Manhattan south of 96th Street.
- Plus tips and any tolls.
- There is no charge for extra passengers, luggage or bags, or paying by credit card.
- The on-screen rate message should read: "Rate #01 – Standard City Rate."
- Make sure to always take your receipt.

Appendix 2

- Description of the categorical variables: we can see the weird cases of number 4 in VendorID and 99 in RatecodeID. Also, the null in trip_type.

VendorID	count
1	4208739
4	196257
2	70364319

RatecodeID	count
99	2526
3	17687
6	390
1	72819739
5	435196
4	70956
2	1422821

trip_type	count
null	354
no	68844062
1	5732115
2	192784

- Weird cases of Datetime data

min(pickup_datetime)	max(pickup_datetime)	min(dropoff_datetime)	max(dropoff_datetime)
2001-01-01 00:02:08	2090-12-31 06:41:26	2001-01-01 01:00:02	2090-12-31 07:18:49

Appendix 3

- Differences in total fare amount for each Borough

Borough_DO	avg(total_amount)
EWR	91.90575069663404
Brooklyn	27.615433490806645
Manhattan	16.510041452563392
Bronx	27.60353242664853
Queens	27.968154936483785
Unknown	30.48763576879663
Staten Island	36.1084090558494

- Description of duration before cleaning

```
+-------+-----------------+
|summary|         duration|
+-------+-----------------+
|  count|         73795576|
|   mean|19.02444278344467|
| stddev|83.81551403619875|
|    min|              0.0|
|    max|43648.01666666667|
+-------+-----------------+
```

Using 3 standard deviation, the maximum values continued high, near 4,5 hours, considering that a straight-line trip between NYC boroughs' borders takes roughly 1 hour after looking on google maps. Therefore, it was assumed as a possibility that a taxi make circles on each trip.

Appendix 4

Variables dropped:

- "ehail_fee" because it had many missing values,
- "av_paid because it was made from the total amount and fare_amount" because it was a mandatory task.
- "pickup_datetime", "dropoff_datetime", "PULocationID", "DOLocationID", "Zone_PU", "Zone_PU"because they were very specific variables and with many categories, respectively.

Appendix 5

- Categorical variables that have different total fare amounts per category

RatecodeID	avg(total_amount)
3	45.28260809904577
5	37.66927490512147
6	10.712174161620762
1	17.164054550549952
4	58.662294953959254
2	68.06670469709013

trip_type	avg(total_amount)
1	15.147609094336952
no	18.472742191931236
2	26.552489302848482

** Trip type 2 is dispatch trips

mta_tax	avg(total_amount)
0.0	36.55852516328567
0.5	18.193904417397956

Appendix 6

- First selection of variables:

'VendorID','passenger_count','trip_distance','RatecodeID','store_and_fwd_flag','payment_type','extra','mta_tax','tip_amount','tolls_amount','improvement_surcharge','total_amount','congestion_surcharge','taxi_type','trip_type','Borough_PU','Borough_DO','year','quarter','dow_long','month','day_time','duration','speed'

- Second selection of variables:

'RatecodeID','tip_amount','trip_distance','duration','tolls_amount','total_amount','Borough_PU','Borough_DO','year','month','speed', 'payment_type'