Hand over report

Project title

**Analysis of New York Taxi Data**

**Big Data Engineering**

Written by

**Declan Stockdale-Garbutt**

Project/Executive Summary

We have been provided the task of analyzing the monthly New York City (NYC) taxi data of the yellow and green taxis from the period of Jan 2019 up to July 2021.

This report is aimed at helping taxi drivers make informed decisions about potential ways to improve their earnings potential. Other aspects of this report may help policy makers and/or regulatory bodies make informed decisions about changes to taxi operations within NYC.

In addition, two machine learning algorithms have been implemented to gain insight into the factors that influence of taxi drivers’ earnings potential.

All analysis was performed using a Databricks notebook utilizing various libraries for data cleaning, querying and machine learning processes.

Project Overview

Purpose

The aim of this project is to help taxi drivers within NYC improve their income where the reported median salary is $44,500 [1]. According to the NYC government website there are a total of 13,587 yellow taxis which predominantly serve the Manhattan area of the city and an additional 18,000 green taxis that operate across the city [2,3].

Definitions

Azure Cloud – Microsoft based cloud computing service

Databricks – A cloud service which allows for analysis of large files in various programming languages

Parquet – Data storage format (similar to csv) which is more efficient to work with

(Py)Spark – Apache Spark product used for analyzing large datasets

Scope

Only the yellow and green taxis are considered for this project. It doesn’t include for hire vehicles or taxi alternatives such as Uber or Lyft. The time frame of this analysis is restricted to from Jan 2019 to July 2021.

Constraints

There are additional limitations imposed during the training of the machine learning algorithms, The main issue is the amount of time needed to train a model on the dataset. Even using only, a fraction of the combined dataset, each model takes a considerable amount of time to train. This limits how many iterations I can perform and limits the amount of optimisation that may lead to an improved final model.

Assumptions

One assumption is that ride sharing services such as Uber or Lyft have had no discernable influence on taxi fares or impacted the number of taxi trips during the timeframe of data collection.

Outstanding issues

Limited training time on models may result in a poorly performing model.

Risks

Any potential errors or incorrect assumptions may negatively impact taxi drivers’ income earnings and may also wrongly influence any policy of infrastructure decision based on this report.

The functionality of the code in the notebook may have to rewritten if updates render it unusable.

Operational Handover

Set up and Analysis

The exact methodology of uploading the data into Azure and loading into Databricks is detailed in these documents [5,6]. It assumes that the maintainer of this project has an Azure account and a Databricks account. Further information is available in the accompanying notebook. If using alternative cloud storage, please seek alternative documentation.

A brief rundown of the process is explained as follows:

1. The datasets can be downloaded online [4] selecting only yellow and green taxis csv from Jan 2019 to July 2021. The taxi\_zone\_lookup csv can also be downloaded.
2. In Azure, create a new container with a name of your choice (storage\_name) and upload the csv files
3. Find your storage account name and key under the Access Keys tab in the Security and Networking section in Azure
4. Login into Databricks and open the side bar and click create followed by notebook giving it a name
5. Initiate a spark session
6. Mount the Azure storage into the spark session

Data Loading

Once the csv’s had been downloaded from the NYC taxi website [4], they were uploaded into the Azure storage account. The csv’s were loaded into Databricks and converted into parquet files and saved. These original files were removed from memory and the parquet files were loaded and used for further processing.

After combining the yellow and green datasets and cleaning it, the final dataset set was saved in parquet format. It’s reloaded at various points throughout the Databricks notebook to save memory.

The taxi\_zone\_lookup csv file can also be uploaded as a standalone csv file as its only used for merging.

Documents for handover

This report is accompanied by a notebook and a html file both detailing the code and analysis.

Data Cleaning

Preparation for merging datasets

There were 124,048,218 rows in the yellow taxi dataset and 8,348,567 in the green dataset for a total of 132,396,785 combined rows.

A data dictionary is also available on the NYC taxi website which explains various aspects of the data which I won’t go into detail here. It was noticed that ehail\_fee and congestion\_surcharge exists in the dataset but not in either data dictionary. Additionally, trip\_type is unique to the green taxi dataset.

The names of the pickup and drop-off timestamps were slightly different and renamed to either pickup\_timestamp or dropoff\_timestamp respectively.

The trip distance was also in miles, so I altered the name to include miles in the columns name for both datasets.

As the yellow dataset lacked ehail\_fee and trip type column, additional columns were added with values set at 0, as there shouldn’t be an ehail fee and the trip type value of 0 can be categorized as ‘unknown’. I added a final column called taxi\_type to distinguish the origin of the row across the dataset.

The column order of the green dataset was rearranged to match the yellow dataset. The schema of both datasets was set according to Appendix Table 1. The datasets were subsequently merged.

Cleaning

The values of the categorical columns were restricted to values in the ranges dictated in the data dictionary, more detail is available in Appendix Table 1. The initial filtering process removed 7,517,105 rows with 124,879,680 remaining. After filtering using the created features, an additional 1,764,636 rows were removed with a final row count of 123,115,044.

Date range

As we’re only concerned about dates ranging from 2019 to 2021, I removed dates that were outside this range. I also decided that trips that started on New Year’s Eve 2018 but finished on Ney Years Day 2019 should be removed as I decided that trips should start within the date range.

Negative values

I removed any negative values associated with numerical columns as they don’t make sense e.g., negative tip, trip\_distance or fare\_amount. A full list can be seen in Appendix Table 1 with columns with integer or float type.

Locations

The pickup and drop-off locations had values ranging from 1-265. Both 265 and 264 were ‘Unknown’. To reduce this, I changed the 265 values to 264.

Cleaning on fees and fares columns

Extra and toll amount

Extra ranged from 0 to hundreds of dollars. There are additional fees associated with airports which may make sense of the higher values. It appeared that values higher than $60 appeared to occur less frequently. I ended up restricting this column to contain values from $0 to $60. There are also high values of toll\_amount which may overlap with airport fee charges. As I can’t determine the difference, l’ve left them in and restricted the tolls amount between $0 and $100.

Tipping

Tip amount is discretionary and varied wildly. I removed rows where the tip was higher than 50% of the total amount. My logic is that it’s not reasonable tip the entirety of the fare. Discretionary tipping in New York is commonplace with values ranging from 10-20% [7]. The maximum of 50% was set to broadly cover any other additional reasons for higher than tipping percentages.

Creating features

Convert trip distance

The trip distance in miles was converted to km by multiplying by 1.608. The miles column was dropped.

Extract information from timestamp

The year, month, day of week and timestamp in seconds was extracted from the pickup and drop off timestamp columns. The length of each trip was found by finding the difference between the newly created drop off and pick up times in seconds. The original timestamp columns were then dropped.

Average speed of trip

The average speed of the trip in km/s was found by dividing the trip distance in km by the number of seconds the trip took. This was multiplied by 3600 to get the final value in km/h.

Cleaning based on created features

Trip time

There appear to be several trip times that last just a few seconds. These may be errors. I set the lowest trip time to 30 seconds as shorter trips wouldn’t be justifiable. There are a range of very long trips up to 16 hours which appear to valid based on other columns. I set the maximum value to 58,363 seconds which equates to 16 hours.

Speed km/h

There were some very slow trips, significantly slower than walking speed. I set the lower limit to 5km/h which is the average walking speed. It might be lower in a busy city with pedestrians and time waiting at pedestrians’ crossings. The maximum speed was set to 120km/h. The logic is that the New York state speed limit is 105km/h, but taxis don’t always obey the speed limit, so I increased it to 120km/h.

Data Analysis

Various analysis has been performed on the dataset relating to various propose questions.

Question 1

a) How many rows in the dataset

The size of the initial dataset was 132,396,785. The cleaning process removed 9,281,741

With a final row count of was 123,115,044.

b) Which day of the week (e.g., Monday, Tuesday etc) had the most trips

From the results of Table 1, Thursday has the highest number of trips. This is likely because Thursday is typically pay day. They would potentially immediately spend their money on pay day or the following day, Friday which is 2nd in the list.

Table

Description automatically generated Table 1. Sum of all trips for each day across cleaned dataset

c) Which hour of the day had the most trips?

Figure 1 shows that 18 or 6pm has the most trips for any hour. Appendix Table 2 shows the entire count of trips per hour.

Chart, bar chart, histogram

Description automatically generated

Figure 1. Bar chart of all trips for each hour

d) What was the average number of passengers?

The average number of passengers was found to be 1.55 or 2 to the nearest individual.

e) What was the average amount paid per trip (using total amount)?

The average total fare is to be $18.49

f) What was the average amount paid per passenger?

The average amount is $11.94 per passenger

For each taxi colour (yellow and green)

1. What was the average, median, minimum, and maximum trip duration in seconds?

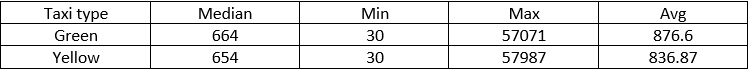


Table 2. Metric of trip durations

1. What was the average, median, minimum, and maximum trip distance in km?

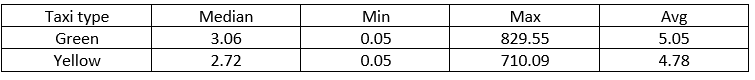


Table 3. Metrics of trip distance in kilometers

1. What was the average, median, minimum, and maximum speed in km per hour?

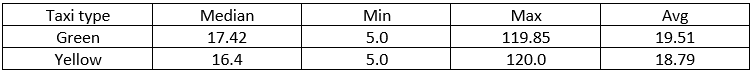


Table 4. Metrics of speed in km/h

Question 3

What was the percentage of trips where the driver received tips?

Drivers received tips 69% of the time.

Question 4

For trips where the driver received tips, what was the percentage where the driver received tips of at least $10

When a driver received a tip, there was a 3.31% chance of it being at least $10.

Question 5

Classifying each trip into bins of trip length 0-5, 5-10,10-20,20-30 and 30+ mins

1. The average speed per bin

Table

Description automatically generated

Table 5. Average speed for each time bin

1. Average distance per dollar (km per $)

Table

Description automatically generated

Table 6. Values of average distance, trip cost and average distance per dollar for bin duration

Question 6

Which duration bin will you advise a taxi driver to target to maximize his income?

As drivers earn money based on fare amount and tips, I added these columns to create an income column. I then found the average income per trip and the average trip length for each bin. As multiple trips in smaller time bins could be completed within the same time as one long trip, I found how many average trips could be completed within the same time under the column (Number of trips in equivalent time). Finally multiplying this value by the average income, we can calculate the maximum potential earnings within a fair time frame for each bin.

Table 7 below clearly shows that trips between 0 and 5 minutes have the most earnings potential.

This analysis assumes that there is no waiting between trips which is unrealistic.

Table

Description automatically generated

Table 7. Various metrics calculated for each time bin. Results are ranked by maximum potential earning income

Machine Learning Modelling

Overview

Two machine learning algorithms have been developed to predict the total amount from given inputs. The main restriction is that fare\_amount can’t be used as it by itself would be a strong indicator, we’re more interested in other potential influential parameters. The possible parameters are those that haven’t been dropped in Appendix Table 1.

Training and testing datasets

The training dataset is limited to the year range 2019 to 2020 using 2021 as the test set. As this is a regression problem, the rmse metric will be used to compare models.

The training dataset contains 109,319,511 rows. This is quite high and will take time to train. To save time, l’ve sampled 0.0001% of that dataset to create a smaller training set of 123,766 rows. Minimal hyperparameter tuning is employed as training still takes a considerable length of time. The total\_amount column is renamed to ‘label’ as it’s the expected input for various pipeline features.

Once the best model has been found using the smaller training set, the model will be trained on the 10% of the original training set.

Random Forest Regression (RFS)

Find most correlated features

Various Pyspark pipelines were used to train a basic RFS model to find the most important columns. These were determined to be: trip\_distance\_km, time\_of\_trip\_seconds, tip\_amount, RatecodeID, tolls\_amount, speed\_km\_h, extra, mta\_tax, DOLocationID, payment\_type. Of these, only RatecodeID, DOLocationID and payment\_type were categorical while the rest were numerical. RatecodeID is connected to extras as it includes airport fees which can dramatically increase the total amount. The money columns make sense as they’re added to find the total amount. The importance of trip distance, time and speed make sense as they factor into the fare\_amount.

Training model based on best features

Data transformation

The columns are restricted to those mentioned above. The input for the machine learning process was processed into a vector.

Hyperparameter tuning

As the training process is quite long even with reduced rows and variables, only a handful of parameter values have been explored. The hyperparameters explored were number of trees (10,15,20) and the maximum tree depth (5, 10, 15). The maximum iterations were set to 20. The small dataset used was further split 80/20 to a train validation set.

The best model had the hyperparameters of 15 trees and a maximum depth of 15.

Model result

10% of the total training rows (up from 0.0001% for tuning) was used to train the final model. The rmse using the previously mentioned hyperparameters was 151.66

Linear model

Data transformation

The same features that were deemed important in the RFR model are used for this model. Various Pyspark transformations are employed to transform the data into the expected format for the model. One hot encoding is performed on the categorical variables followed by assembling a vector of the one hot encoded variable and the numeric variables.

Hyperparameter tuning and validation

Only two hyperparameters were investigated while being trained on 10% of the total training dataset. The first was the regularization parameter with values 0.2, 0.4, 0.8. The second was the elastic net parameter with values 0.0, 0.5, 1.0. The same 80/20 train validation split was used.

The best model had a regularization of 0.4 and the elastic net was 0.

Model result

The same 10% of data was used to train the RFS model using the above parameters. The final rmse was 152.52

Testing best model (Random Forest Regression) on 2021 data

Final rmse of 125.63 was achieved which is noticeable lower compared to both the linear model and the random forest regression model. This likely indicates that outliers exist within the training set. The predictions diverge as the total\_amount increase which can be seen in Figure 2 and I s more apparent in Appendix Figure 2. Training on a dataset with more values within this range would potentially improve the overall model.

Chart, scatter chart

Description automatically generated

­­­­

­

Figure 2. Output of best RFR model on 0.0005% of the test dat. The red line indicates a perfect prediction

Conclusions

Various aspects of the New York City taxi datasets have been explored and analysed to help drivers increase their earnings potential. The strongest indicators or the final fare were found to be trip distance and length. Additional important factors include the RatecodeID which includes additional airport fees and tip amount. Two machine learning algorithms were implemented to predict total amount. The best performing model achieved a test score of 125.63 which is poor and the model should not be relied upon.

Limitations caused by training on the large dataset prohibited the extent to which various models could be explored and tuned. Future investigations should address these needs to improve model predictive power. Further improvements may also be made by limiting the analysis to trips that started and ended within the same boroughs.

1. References
2. <https://www.salary.com/research/salary/benchmark/taxi-driver-salary/new-york-ny>
3. <https://www1.nyc.gov/site/tlc/businesses/yellow-cab.page>
4. Morris, C. Zawadi (December 21, 2011). "Cuomo Signs 5-Boro Taxi Plan Into Law". Patch. Retrieved December 18, 2013.https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
5. Lab 4 Exercise 1 <https://colab.research.google.com/drive/1avfVLJ2aoxLhUZwqFvUKqxienDhm-vR5#scrollTo=MF-QvXxdGNTn>
6. Lab 4 Exercise 2 <https://colab.research.google.com/drive/1Vcst_2QHZxO8zxvbDp6NGV1hW2two_FK#scrollTo=hYFU1gWzvaT>
7. Taxi tipping https://www.tripsavvy.com/guide-to-tipping-in-new-york-city-4177115#:~:text=Taxi%20drivers%20should%20be%20tipped,being%20carried%20in%20the%20shuttle.
8. Appendix

Appendix Table 1

|  |  |  |  |
| --- | --- | --- | --- |
| Origin | Name | Converted datatype | Comment |
| Y/G | VendorID | integer | 1 = Creative Mobile Technologies  2 = VeriFone Inc. |
| Y/G | pickup\_timestamp | timestamp | Dropped after extracting information |
| Y/G | dropoff\_timestamp | timestamp | Dropped after extracting information |
| Y/G | passenger\_count | integer | Final range 1-9 |
| Y/G | trip\_distance | float | Value in miles. Dropped after converting to km |
| Y/G | RatecodeID | integer | 1= Standard rate  2=JFK  3=Newark  4=Nassau or Westchester  5=Negotiated fare 6=Group ride |
| Y/G | store\_and\_fwd\_flag | string | Seems like a connection issue (irrelevant) |
| Y/G | PULocationID | integer | Coded pick up location Range 1-264 |
| Y/G | DOLocationID | integer | Coded dropoff location Range 1-264 |
| Y/G | payment\_type | integer | 1= Credit card  2= Cash  3= No charge  4= Dispute  5= Unknown  6= Voided trip |
| Y/G | fare\_amount | float | Cost of metered fare |
| Y/G | extra | float | Rush hour or overnight fee |
| Y/G | mta\_tax | float | Standard $0.50 tax |
| Y/G | tip\_amount | float | Amount that was tipped |
| Y/G | tolls\_amount | float | Tolls paid |
| Y/G | improvement\_surcharge | float | Minimum $0.30c |
| Y/G | total\_amount | float | Sum of all other fares, taxes and tip |
| Y/G | congestion\_surcharge | float | Fee for congestion |
| G | ehail\_fee | float | Fee paid for electronic ehail |
| G | trip\_type | string | 1= Street-hail  2= Dispatch |
| Created | pickup\_timestamp | timestamp | Time in seconds of pickup. Dropped after extracting information |
| Created | dropoff\_timestamp | timestamp | Time in seconds of drop off. Dropped after extracting information |
| Created | taxi\_type | string | Yellow or green |
| Created | trip\_distance\_km | float | Converted miles to km |
| Created | year | integer | Extracted from original timestamp |
| Created | month | integer | Extracted from original timestamp |
| Created | day\_of\_week | integer | Extracted from original timestamp  Starts at 1 = Sunday |
| Created | 24\_hour | integer | Extracted from original timestamp |
| Created | time\_of\_trip\_seconds | long | Subtract drop off from pickup timestamp in seconds |
| Created | speed\_km\_h | float | Divide trip length in km by seconds and multiply by 3600 |
| Merged | PU\_Borough | String | Merged from taxi\_location csv - not used |
| Merged | DO\_Borough | String | Merged from taxi\_location csv – not used |

Appendix Table 1. Summary of columns containing information about where they originated and any created columns. Y = yellow taxi dataset, G = Green taxi dataset

Appendix Table 2

|  |  |
| --- | --- |
| Hour (24 Hour time) | Number of trips |
| **18** | **8328474** |
| 17 | 7722257 |
| 19 | 7660937 |
| 15 | 7345020 |
| 14 | 7242781 |
| 16 | 6983770 |
| 13 | 6794193 |
| 20 | 6747293 |
| 12 | 6680089 |
| 21 | 6508282 |
| 11 | 6263958 |
| 22 | 5969997 |
| 10 | 5945432 |
| 9 | 5792968 |
| 8 | 5632445 |
| 23 | 4685752 |
| 7 | 4349092 |
| 0 | 3313657 |
| 6 | 2424067 |
| 1 | 2264399 |
|  |  |

Appendix Table 2. Number of trips for each hour across entire cleaned dataset. Ordered by decreasing number of trips.

Appendix Figure 1

Table

Description automatically generated

Appendix Figure 2. Feature importance for all input columns of basic random forest regression model. Higher values mean a high importance and more predictive power.

Appendix Figure 2

Chart, scatter chart

Description automatically generated

Appendix figure 3. Plot of predictions and actual values on 0.05% of the test data equating to almost 690,000 rows. Line indicates perfect predictions