Analyzing New York City Taxi Data

LE Thi Hoa - 12310380
TRAN Hai Linh - 12310487

October 2023

I. Introduction

This project aims to analyze and gain a better understanding of taxi trip data in NYC

II. Data Selection

1. Sample NYC Data

The taxi dataset used in this project is a subset of taxi trip data in NYC. This dataset consists of 99999 taxi trips, each with 12 attributes

```
root
 |-- medallion: string (nullable = true)
 |-- hack_license: string (nullable = true)
 |-- vendor_id: string (nullable = true)
 |-- rate_code: string (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- pickup_datetime: string (nullable = true)
 |-- dropoff_datetime: string (nullable = true)
 |-- passenger_count: string (nullable = true)
 |-- pickup_longitude: string (nullable = true)
 |-- pickup_latitude: string (nullable = true)
 |-- dropoff_longitude: string (nullable = true)
 |-- dropoff_latitude: string (nullable = true)
```

Figure 1: Schema of the data

Basic information about the attributes of data:

+ **Medallion**: The cab (a hashed version of the medallion number)
+ **Hack_license**: The driver (a hashed version of the hack license)
+ **vendor_id**: a code indicating the provider associated with the trip record
+ **Rate_code**: A rate code is a numeric code or an option code used to determine the pricing structure for a taxi trip

+ **store_and_fwd_flag**: This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server—Y=store and forward; N=not a store and forward trip.
+ **pickup_datetime**: date and time when the meter was engaged
+ **dropoff_datetime**: date and time when the meter was disengaged
+ **passenger_count**: the number of passengers in the vehicle (driver entered value)
+ **pickup_longitude**: the longitude where the meter was engaged
+ **pickup_latitude**: the latitude where the meter was engaged
+ **dropoff_longitude**: the longitude where the meter was disengaged
+ **dropoff_latitude**: the latitude where the meter was disengaged

2. NYC Boroughs

This is the boundary data for each borough in geojson format. This dataset is used to identify the names of the boroughs for both pick-up and drop-off locations of each taxi trip

Basic information about the attributes of data:

+ **Geometry**: Geometry represents geometric objects in space, including the longitude and latitude points of areas within New York City
+ **boroughCode**: Code of borough
+ **Borough**: Name of borough

III. Preprocessing

1. Feature engineering

Feature engineering is the process of transforming raw data into useful features to maximize the potential of the data.

- For NYC boroughs data:
- This dataset includes 104 zones corresponding to the 5 boroughs of New York City, which are:
    + Manhattan - 29 zones
    + Bronx - 30 zones
    + Brooklyn - 21 zones
    + Queens - 20 zones

    + Staten Island - 4 zones
- We sort the data in descending order of the total area of each borough. Then, for each borough, we further sort the zones in descending order of the area of each zone. This data is relatively small. So, We broadcast it to the different workers. The results obtained are as follows:
    + The largest is **Queens** borough with a total area of 283354532.24758565
    + Then the borough of **Brooklyn** with total area: 182049468.87353534
    + The **Staten Island** borough with total area: 150892754.55717745
    + The **Bronx** borough with total area: 110294139.76927057
    + The **Manhattan** borough with total area: 59119007.27194083

- For Sample NYC data:
- We create a "pickup_ts" column and "dropoff_ts" column. The we used the unix_timestamp function to convert the pickup_datetime and dropoff_datetime into milliseconds since epoch time of Unix systems
- We create a "**duration**" column to store the time it takes for a trip, calculated from the moment of pickup to the drop-off (dropoff_ts - pickup_ts)
- We use the Shapely library to perform a search for the borough names of pickup and drop-off points when given their longitude and latitude coordinates (longtitude_pickup, latitude_pickup, longtitude_dropoff, latitude_dropoff). We define a user-defined function (UDFs) named **'get_borough'** to perform this. For locations whose names are not found in the NYC boroughs, they will be designated as **Unknown**

2. Data Cleaning

We need to clean the data, removing inappropriate records and outliers. We have performed the data cleaning steps as follows:

- Check the null values for each column to determine how many missing values are in each column. The results show:
    + Number of null values in medallion column: 0
    + Number of null values in hack_license column: 0

    + Number of null values in vendor_id column: 0
    + Number of null values in rate_code column: 0
    + Number of null values in store_and_fwd_flag column: 99626
    + Number of null values in pickup_datetime column: 0
    + Number of null values in dropoff_datetime column: 0
    + Number of null values in passenger_count column: 0
    + Number of null values in pickup_longitude column: 0
    + Number of null values in pickup_latitude column: 0
    + Number of null values in dropoff_longitude column: 0
    + Number of null values in dropoff_latitude column: 0
- Most of the data in column '**store_and_fwd_flag**' is missing, so we decided to drop this column.
- We proceeded to check for duplicate records, and the result showed that there are no duplicate records.
- Each taxi, when performing a trip, must have one or more passengers. Therefore we check and remove records with less than 1 passenger (namely 0)
- We expect that drop-off times of trips occur after their respective pick-up times, and the duration of each trip doesn't exceed 4 hours. So, we will remove records with negative duration and those exceeding 4 hours (14400000 milliseconds)
- Remove records with pickup and drop-off locations not within the 5 boroughs of NYC (**Unknown**)
- In each driver's case, we expect that the pickup time of a trip must occur after the drop-off time of the preceding trip. This means a trip must be completed before another one can start. Therefore, we sort the trips for each driver by their start time and then remove the records where the pickup time occurs before the drop-off time of the preceding trip

IV. Analysis

1. **Utilization**

This is per taxi/driver. This can be computed by computing the idle time per taxi. We calculate the waiting time for each trip by each driver using a window operation. (pickup_ts - previous)

Then we group by hack_license to calculate the total waiting time for each driver (milliseconds).

We also group by the drop-off borough (dropoff_borough) and then calculate the total waiting time in each borough (milliseconds). The results are as follows:

dropoff_borough	sum(idle_time)
Queens	7409880000
Brooklyn	4104480000
Staten Island	3840000
Manhattan	77679300000
Bronx	447660000

Table 1: Result of calculating the total idle time in each borough

2. **Compute the average time**

Compute the average time it takes for a taxi to find its next fare(trip) per destination borough. We group by each borough and calculate the average waiting time. The results are as follows:

dropoff_borough	average_wait_time
Queens	1733710.8095460928
Brooklyn	1615301.0625737899
Staten Island	1280000.0
Manhattan	983083.1730282474
Bronx	1639780.2197802197

Table 2: The average idle time in each borough

3. **The number of the trips:**

- The number of trips that started and ended within the same borough
- The number of trips that started in one borough and ended in another one

=> Result:

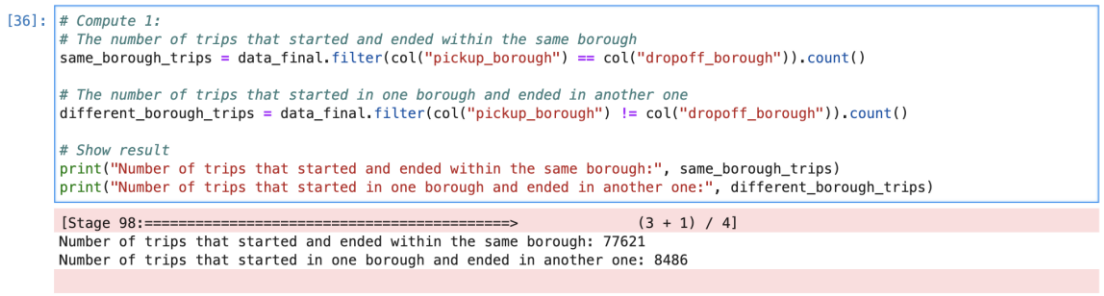

Figure 2: Result of computing the number of the trips

Code - Link project:

https://github.com/LeHoa98ptit/Spark_Project_Lyon_1/tree/main