

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM**

**KHOA TOÁN – TIN HỌC**

....📖....

## **BÁO CÁO CUỐI KỲ**

**Môn Python cho khoa học dữ liệu – MTH10605**

**Học kỳ I (2022 - 2023)**

# **ĐỀ TÀI: PHÂN TÍCH VÀ DỰ ĐOÁN THỜI GIAN DI CHUYỂN CỦA CÁC CHUYẾN TAXI TẠI THÀNH PHỐ NEW YORK**

Nhóm thực hiện: Nhóm 17

Tên thành viên:

1. Huỳnh Quang Trung – 20280108 (Nhóm trưởng)
2. Hồ Ngọc Ân – 20280001
3. Trần Tuấn Thái – 20280082
4. Hòa Ngọc Tú – 20280111

Giảng viên hướng dẫn: ThS. Hà Văn Thảo.

**TP. Hồ Chí Minh, ngày 14 tháng 01 năm 2023**

# MỤC LỤC

<b>MỞ ĐẦU .....</b>	<b>4</b>
<b>NỘI DUNG.....</b>	<b>5</b>
<b>1. Giới thiệu.....</b>	<b>5</b>
<b>1.1. Giới thiệu chung về đề tài.....</b>	<b>5</b>
<b>1.2. Mục tiêu của đề tài. ....</b>	<b>5</b>
<b>2. Nội dung thực hiện. ....</b>	<b>5</b>
<b>2.1. Cài đặt và gọi các thư viện cần thiết.....</b>	<b>5</b>
<b>2.2. Tìm hiểu và khám phá bộ dữ liệu.....</b>	<b>6</b>
2.2.1. Thông tin về bộ dữ liệu. ....	6
2.2.2. Dữ liệu có các dòng bị lặp không?.....	7
2.2.3. Các thuộc tính mang kiểu dữ liệu gì? .....	7
<b>2.3. Tiền xử lý và làm sạch dữ liệu.....</b>	<b>9</b>
2.3.1. Chuyển cột <b>pickup_datetime</b> và <b>dropoff_datetime</b> sang kiểu dữ liệu <b>date_time</b> . ....	9
2.3.2. Chuyển cột <b>store_and_fwd_flag</b> sang kiểu dữ liệu <b>bool</b> .....	9
2.3.3. Chuyển cột <b>id</b> sang kiểu dữ liệu <b>string</b> . ....	9
2.3.4. Thêm cột <b>distance</b> vào tập dữ liệu. ....	9
2.3.5. Thêm cột <b>speed</b> vào tập dữ liệu. ....	10
<b>2.4. Khám phá dữ liệu (tiếp tục). ....</b>	<b>11</b>
2.4.1. Khảo sát cột <b>passenger_count</b> . ....	11
2.4.2. Khảo sát cột <b>trip_duration</b> và <b>distance</b> . ....	12
2.4.3. Khảo sát cột <b>speed</b> . ....	19
<b>2.5. Tiền xử lý dữ liệu (tiếp tục). ....</b>	<b>21</b>
2.5.1. Xóa các bản ghi có <b>passenger_count = 0, 7, 8, 9</b> .....	21

2.5.2.	Xoá các chuyến có trip_duration lớn hơn 82600. ....	21
2.5.3.	Xoá các chuyến có distance = 0 và trip_duration lớn hơn 120 giây. ....	21
2.5.4.	Xoá các chuyến có tốc độ di chuyển từ 100km/h trở lên.....	21
<b>3.</b>	<b>PHÂN TÍCH DỮ LIỆU. ....</b>	<b>24</b>
3.1.	Câu hỏi phân tích số 1: trip_duration thay đổi như thế nào theo các ngày .....	24
	trong tuần và các giờ trong ngày? .....	24
3.2.	Câu hỏi phân tích số 2: Có gì đáng chú ý ở vị trí điểm đón và trả khách của các chuyến đi hay không? .....	34
<b>4.</b>	<b>HUẤN LUYỆN MÔ HÌNH DỰ ĐOÁN TỔNG THỜI GIAN DI CHUYỂN CỦA CHUYẾN ĐI (TRIP_DURATION).....</b>	<b>38</b>
4.1.	Tiền xử lý.....	38
4.2.	Kết quả. ....	40
<b>TỔNG KẾT.....</b>		<b>41</b>

## MỞ ĐẦU

Học phần Python cho khoa học dữ liệu – MTH1065 trình bày các kiến thức cơ sở về lập trình Python cho khoa học dữ liệu phục vụ cho xử lý dữ liệu tín hiệu đa chiều trong phân loại dữ liệu hay nhận dạng đối tượng, khai thác dữ liệu, phân tích dữ liệu, thống kê, máy học,... Nội dung chính của chương trình sẽ trình bày cơ sở để sinh viên có khả năng lập trình bằng Python trong khoa học dữ liệu. Sau đó sinh viên sẽ áp dụng các kiến thức để giải quyết những bài tập lớn.

Đề tài kết thúc môn học lần này nhằm giúp sinh viên nắm được và vận dụng được các kỹ thuật lập trình, các thuật toán đã được tìm hiểu trước đó để tiến hành các bước phân tích dữ liệu, tạo giả thuyết, lấy dữ liệu, tiền xử lý, phân tích, đánh giá chất lượng, đưa ra các phán đoán. Với đề tài “Phân tích và dự đoán thời gian di chuyển của các chuyến taxi tại thành phố New York – New York City Taxi Trip Duration”, nhóm chúng em mong rằng sẽ chia sẻ được những thông tin hữu ích, đóng góp vào bộ tài nguyên cho ngành học của chúng ta thêm phong phú hơn, làm nguồn tham khảo cho tất cả mọi người.

Nhóm sinh viên thực hiện đề tài.

# NỘI DUNG

## 1. GIỚI THIỆU.

### 1.1. Giới thiệu chung về đề tài.

Trong bài báo cáo này, nhóm sẽ sử dụng bộ dữ liệu về thông tin các chuyến taxi tại New York vào 6 tháng đầu năm 2016 dựa trên dữ liệu [2016 NYC Yellow Cab trip record data](#) có sẵn trên BigQuery của Google Cloud Platform. Dữ liệu gốc được phát hành bởi NYC *Taxi and Limousine Commission (TLC)* và có sẵn trên trang [NYC TLC](#). Tập dữ liệu được dùng trong bài báo cáo này đã được lấy mẫu (sample) và tiền xử lý để sử dụng cho cuộc thi [New York City Taxi Trip Duration](#) trên Kaggle vào năm 2017.

### 1.2. Mục tiêu của đề tài.

Xây dựng một mô hình dự đoán tổng thời gian di chuyển của một chuyến taxi tại thành phố New York dựa vào các biến như thời gian đón, tọa độ địa lý, số lượng hành khách và một số biến số khác.

Sau khi thực hiện xong đề tài, sinh viên nắm vững các kiến thức cơ bản về lập trình Python và rèn luyện được kỹ năng phân tích, áp dụng các thuật toán vào thực tế.

## 2. NỘI DUNG THỰC HIỆN.

### 2.1. Cài đặt và gọi các thư viện cần thiết.

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import geopy.distance
import seaborn as sns
plt.style.use('ggplot')

import pickle
from sklearn.model_selection import train_test_split
import xgboost as xgb
```

## 2.2. Tìm hiểu và khám phá bộ dữ liệu.

### 2.2.1. Thông tin về bộ dữ liệu.

Tại đây, chúng ta sẽ sử dụng hàm `read_csv` của thư viện Pandas để load dữ liệu. Hai tập dữ liệu `train.csv` và `test.csv` được upload lên Github nhằm mục đích có thể đọc được tập dữ liệu thông qua link raw data, giúp linh hoạt hơn trong việc chạy được file notebook trong nhiều trường hợp khác nhau mà không phụ thuộc vào việc phải cần có file dữ liệu tại local.

Output[1]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053

Output[2]:

Tập dữ liệu train bao gồm 1458644 dòng và 11 cột.

Quan sát bộ dữ liệu thu thập được ở trên, ta có thể thấy có 11 cột với tổng cộng 1458644 dòng theo các quan sát trong tập dữ liệu. Mặc dù tập dữ liệu test chỉ có 9 thuộc tính, tập dữ liệu train có 11 thuộc tính, trong đó 1 thuộc tính là target, còn lại 1 thuộc tính không có trong tập test là `dropoff_datetime`. Chúng em quyết định không loại cột này ra khỏi tập dữ liệu train để cung cấp tập mở rộng các biến và có thể sử dụng tùy ý trong quá trình làm việc.

## Giải thích rõ ràng hơn về ý nghĩa của các thuộc tính:

Tập dữ liệu test.csv đã xử lý gồm 11 cột, cột cuối cùng (trip\_duration) là giá trị cần được dự đoán:  
**id** - một mã định danh duy nhất cho mỗi chuyến đi  
**vendor\_id** - mã cho biết nhà cung cấp được liên kết với hồ sơ chuyến đi (vì thông tin không được thu thập và cung cấp bởi TLC mà được cung cấp bởi 2 nhà cung cấp khác nhau được đánh số là 1 và 2)  
**pickup\_datetime** - Ngày và giờ khi đồng hồ của taxi bắt đầu tính số mét  
**dropoff\_datetime** - ngày và giờ khi kết thúc tính số mét  
**passenger\_count** - số hành khách trên phương tiện (tài xế nhập thông tin này)  
**pickup\_longitude** - kinh độ nơi bắt đầu tính số kilomet  
**pickup\_latitude** - vĩ độ nơi bắt đầu tính số mét  
**dropoff\_longitude** - kinh độ nơi kết thúc tính số kilomet  
**dropoff\_latitude** - vĩ độ nơi kết thúc tính số mét  
**store\_and\_fwd\_flag** - đánh dấu 1 bản ghi có được ghi lại trong bộ nhớ của phương tiện trước khi được gửi đến cho nhà cung cấp không (điều này xảy ra vì phương tiện không có kết nối đến máy chủ khi di chuyển)  
**trip\_duration** - thời gian của chuyến tính bằng giây

### 2.2.2. Dữ liệu có các dòng bị lặp không?

Vì dữ liệu ở cột id là duy nhất cho mỗi bản ghi, nên vì vậy ta sẽ tiến hành kiểm tra xem có bản ghi nào trùng lặp với nhau trong tập dữ liệu này hay không thông qua cách kiểm tra sự trùng lặp dữ liệu ở cột **id**.

Output[3]:

Không có dòng nào trùng lặp trong tập dữ liệu

Do tập dữ liệu đã được Ban tổ chức tiền xử lý trước để dùng trong cuộc thi nên trong tập dữ liệu của chúng ta không có các dòng dữ liệu bị trùng lặp với nhau.

### 2.2.3. Các thuộc tính mang kiểu dữ liệu gì?

Sử dụng info() để xem thông tin chi tiết của các cột có trong dataframe, mục đích chính ở đây sẽ là xem kiểu dữ liệu của từng cột.

Output[4]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1458644 entries, 0 to 1458643
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                    1458644 non-null object
1   vendor_id            1458644 non-null int64
2   pickup_datetime      1458644 non-null object
3   dropoff_datetime     1458644 non-null object
4   passenger_count      1458644 non-null int64
5   pickup_longitude     1458644 non-null float64
6   dropoff_latitude     1458644 non-null float64
7   dropoff_longitude    1458644 non-null float64
8   dropoff_latitude     1458644 non-null float64
9   store_and_fwd_flag   1458644 non-null object
10  trip_duration        1458644 non-null int64
dtypes: float64(4), int64(3), object(4)
memory usage: 122.4+ MB
```

Ở đây, có 4 thuộc tính mang kiểu dữ liệu object là id, pickup\_datetime, dropoff\_datetime và store\_and\_fwd\_flag. Ta tiếp tục tiến hành xem các kiểu dữ liệu có trong các thuộc tính này.

Output[6]:

```
{<class 'str'>}
{<class 'str'>}
{<class 'str'>}
{<class 'str'>}
```

Ta thấy, tất cả các phần tử tại các thuộc tính này đều mang kiểu dữ liệu là string. Để tiếp tục bài toán, tại đây ta có thể đưa ra các hướng giải quyết như sau:

- Đối với id: Đây là một mã định danh cho mỗi chuyến taxi nên ta có thể giữ nguyên kiểu dữ liệu của nó là string.
- Đối với pickup\_datetime và dropoff\_datetime: Ta sẽ chuyển chúng sang kiểu dữ liệu date\_time để phù hợp với ngữ cảnh dữ liệu mà nó lưu trữ và thuận tiện để làm việc với 2 cột này.
- Đối với store\_and\_fwd\_flag: Đây là cột mang ý nghĩa phân loại và đánh dấu, ta sẽ chuyển nó sang kiểu bool.



## 2.3. Tiền xử lý và làm sạch dữ liệu.

Dựa vào các quyết định ở phía trên, đầu tiên ta sẽ:

2.3.1. Chuyển cột *pickup\_datetime* và *dropoff\_datetime* sang kiểu dữ liệu *date\_time*.

2.3.2. Chuyển cột *store\_and\_fwd\_flag* sang kiểu dữ liệu *bool*.

2.3.3. Chuyển cột *id* sang kiểu dữ liệu *string*.

Ta xem lại kiểu dữ liệu của các cột sau khi được xử lý

Output[10]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1458644 entries, 0 to 1458643
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id                    1458644 non-null object
1   vendor_id             1458644 non-null object
2   pickup_datetime       1458644 non-null datetime64[ns]
3   dropoff_datetime      1458644 non-null datetime64[ns]
4   passenger_count       1458644 non-null int64
5   pickup_longitude      1458644 non-null float64
6   pickup_latitude       1458644 non-null float64
7   dropoff_longitude     1458644 non-null float64
8   dropoff_latitude      1458644 non-null float64
9   store_and_fwd_flag    1458644 non-null bool
10  trip_duration         1458644 non-null int64
dtypes: bool(1), datetime64[ns](2), float64(4), int64(2), object(2)
memory usage: 112.7+ MB
```

2.3.4. Thêm cột *distance* vào tập dữ liệu.

Dựa vào tọa độ điểm đón và trả khách được cung cấp trong tập dữ liệu, ta sẽ tiến hành tính khoảng cách di chuyển của chuyến xe bằng cách sử dụng thư viện hỗ trợ là *geopy*. Khoảng cách tính được ở đơn vị *km*.

Theo tài liệu của thư viện *Geopy* thì *geopy.distance.distance* sử dụng khoảng cách trắc địa (*geodesic distance* - khoảng cách ngắn nhất trên bề mặt của mô hình hình elip của trái đất) để tính toán khoảng cách giữa 2 điểm. Tìm hiểu thêm về cách tính khoảng cách của thư viện *Geopy* [tại đây](#).

Output[14]:

up_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration	distance
40.767937	-73.964630	40.765602	False	455	1.502172
40.738564	-73.999481	40.731152	False	663	1.808660
40.763939	-74.005333	40.710087	False	2124	6.379687
40.719971	-74.012268	40.706718	False	429	1.483632
40.793209	-73.972923	40.782520	False	435	1.187038

### 2.3.5. Thêm cột speed vào tập dữ liệu.

Ở đây chúng ta sẽ tính tốc độ của chuyến đi thông qua công thức tính vận tốc đơn giản là quãng đường/thời gian:  $V = S/t$  và quy về đơn vị km/h.

Output[18]:

dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration	distance	speed
-73.964630	40.765602	False	455	1.502172	11.885316
-73.999481	40.731152	False	663	1.808660	9.820778
-74.005333	40.710087	False	2124	6.379687	10.813029
-74.012268	40.706718	False	429	1.483632	12.450063
-73.972923	40.782520	False	435	1.187038	9.823760

## 2.4. Khám phá dữ liệu (tiếp tục).

Ở phần này, ta sẽ bắt đầu tìm hiểu riêng một số thuộc tính của tập dữ liệu.

Output[21]:

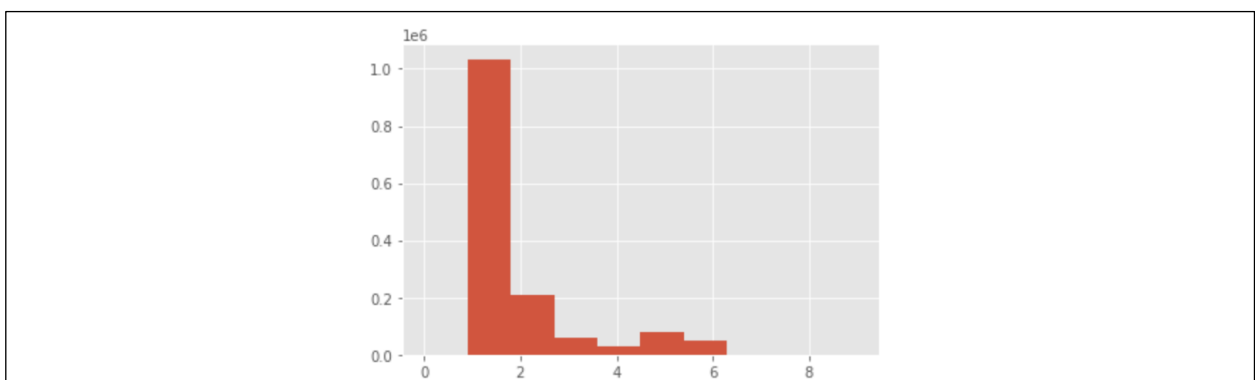
	<b>passenger_count</b>	<b>pickup_datetime</b>	<b>dropoff_datetime</b>	<b>pickup_latitude</b>
<b>missing_ratio</b>	0.0	0.0	0.0	0.000000
<b>min</b>	0.0	2016-01-01 00:00:17	2016-01-01 00:03:31	34.359695
<b>max</b>	9.0	2016-06-30 23:59:39	2016-07-01 23:02:03	51.881084

Dựa vào bảng kết quả trên ta có nhận xét như sau:

- Tất cả các thuộc tính đều không có giá trị NA (giá trị bị thiếu).
- Có chuyến đi với số lượng hành khách thấp nhất là 0 và cao nhất là 9 người. Chúng ta sẽ tìm hiểu vấn đề này trong phần tiếp theo.
- Có chuyến đi với thời gian di chuyển chỉ 1 giây với quãng đường di chuyển là 0 km. Đây là chuyến đi với thời gian ngắn bất thường.
- Chúng ta có các chuyến đi với độ dài trip\_duration lên tới 3526282 giây, tương ứng với hơn 40 ngày. Có chuyến đi với quãng đường lên tới 1240.5 km, đặc biệt hơn còn có chuyến đi có tốc độ lên tới 9279.58 km/h. Ngoài ra, chúng ta còn có các chuyến đi có vẻ không di chuyển với quãng đường và tốc độ bằng 0. Đây đều là những chuyến đi có thuộc tính rất bất thường.

### 2.4.1. Khảo sát cột **passenger\_count**.

Output[22]:



Output[23]:

```
1    1033540
2     210318
5      78088
3      59896
6      48333
4      28404
0         60
7          3
9          1
8          1
Name: passenger_count, dtype: int64
```

Nhận xét:

- Ta thấy phần lớn các chuyến đi sẽ có số lượng là 1 hành khách với 1033540 bản ghi, tiếp sau đó là 2 người với 210318 bản ghi. Đối với các chuyến đi có từ 3 đến 6 hành khách cũng có số lượng nhất định nằm trong khoảng từ 30000 đến 80000 bản ghi.
- Ta cũng thấy có 60 chuyến đi với số lượng là 0 hành khách, đối với các chuyến có 7, 8 và 9 hành khách cũng có rất ít bản ghi.

Vì tồn tại các dòng dữ liệu với các số liệu bất thường, đồng thời chúng cũng chiếm phần thiểu số trong tập dữ liệu, nên ta quyết định sẽ xóa bỏ các dòng có số lượng hành khách là 0, 7, 8 và 9 ra khỏi tập dữ liệu.

#### 2.4.2. Khảo sát cột *trip\_duration* và *distance*.

Output[24]:

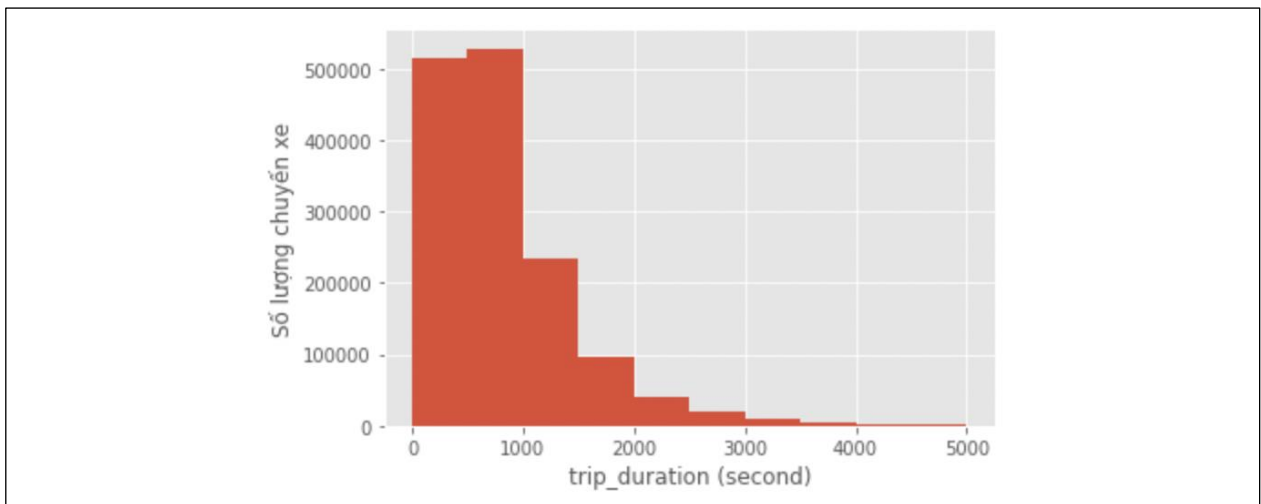
```
978383    3526282
924150    2227612
680594    2049578
355003    1939736
1234291     86392
295382     86391
73816      86390
59891      86387
1360439    86385
753765     86379
1221666    86378
91717      86378
1138915    86377
66346      86377
1284067    86369
Name: trip_duration, dtype: int64
```

Nhận xét:

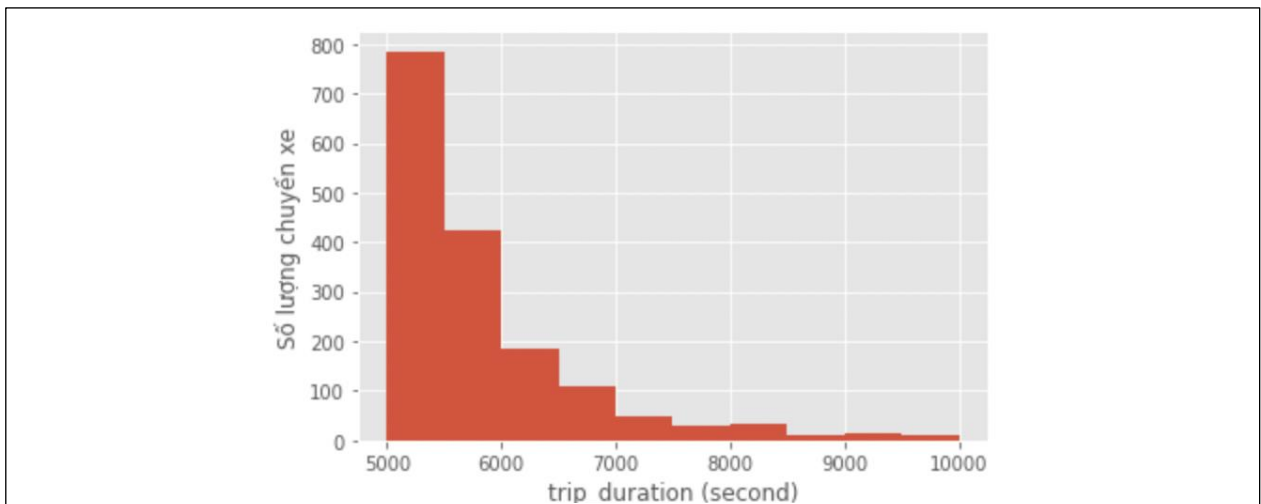
- Có 4 chuyến đi với thời lượng di chuyển khá lớn từ gần 2 triệu giây đến hơn 3.5 triệu giây (kéo dài hơn 40 ngày).
- Ngoài 4 chuyến đi với thời lượng di chuyển rất lớn ở phía trên, thì các chuyến đi còn lại trong tập dữ liệu sẽ có trip\_duration không vượt quá 86400 giây.

Tiếp theo, ta sẽ quan sát sự phân bố về số lượng các chuyến taxi có trip\_duration nằm trong các khoảng 0-5000 giây, 5000-10000 giây, 10000-80000 giây, 80000-82000 giây, 82000-83000 giây, 83000-84000 giây và 84000-84600 giây.

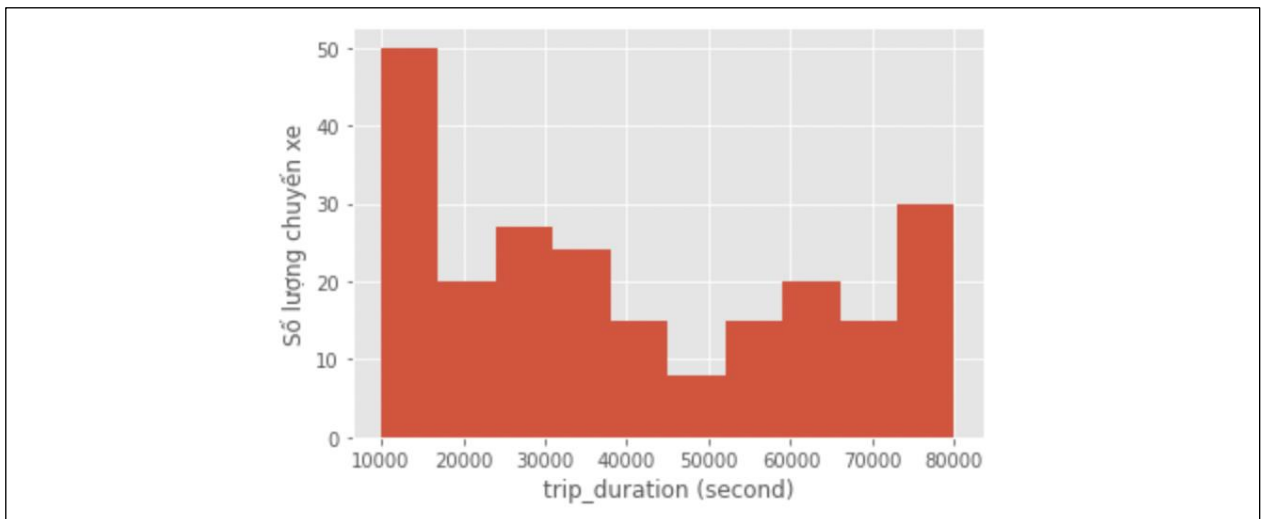
Output[25]:



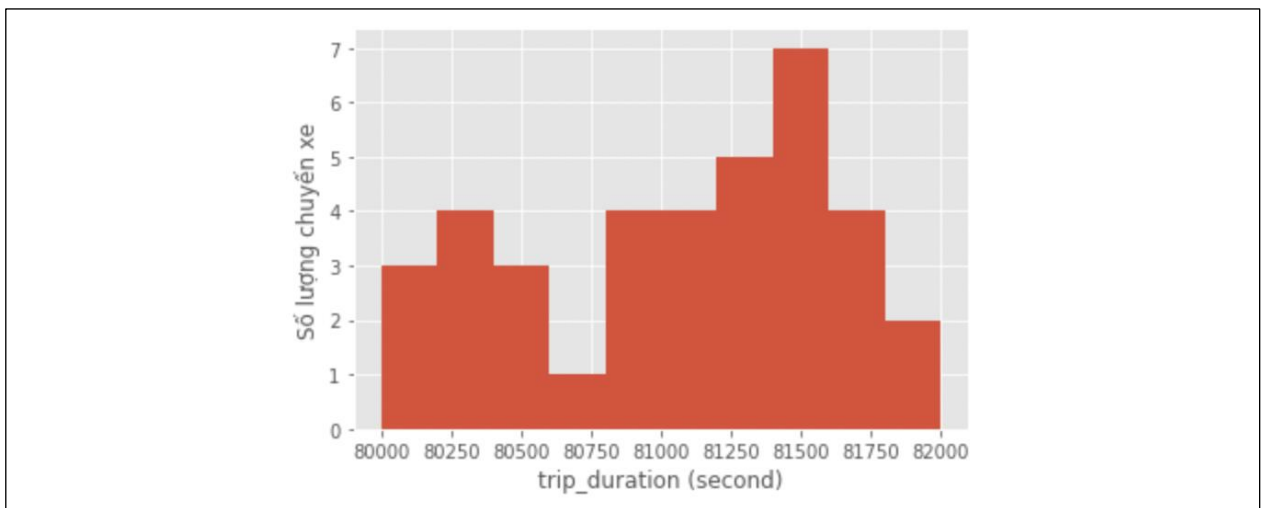
Output[26]:



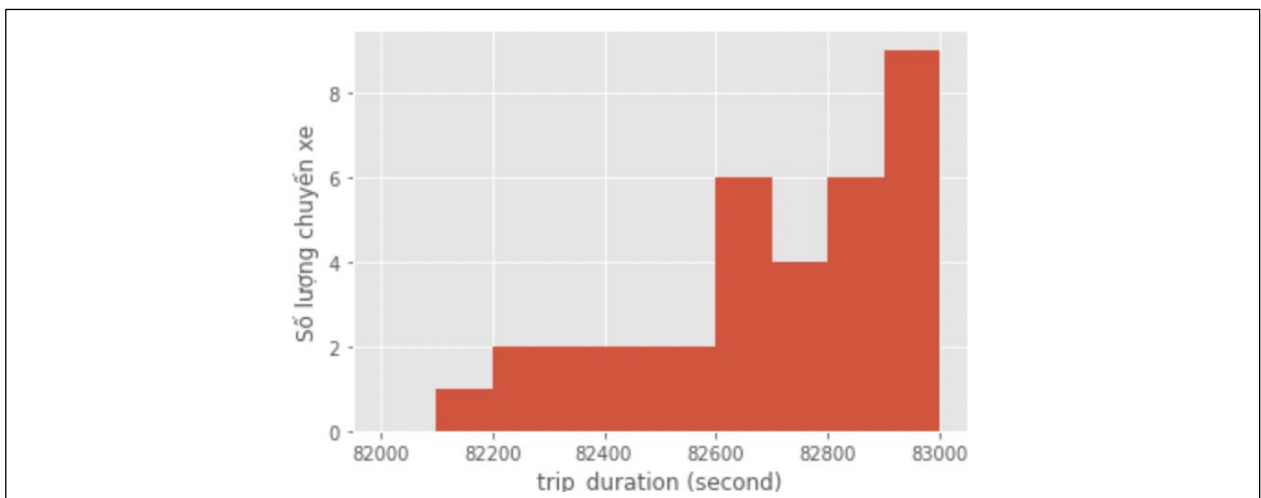
Output[27]:



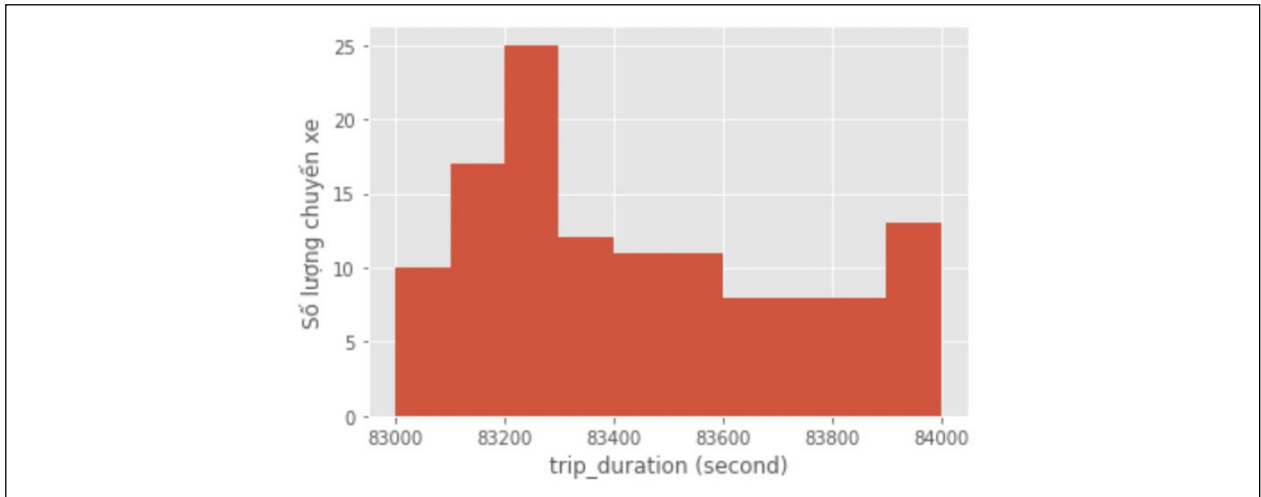
Output[28]:



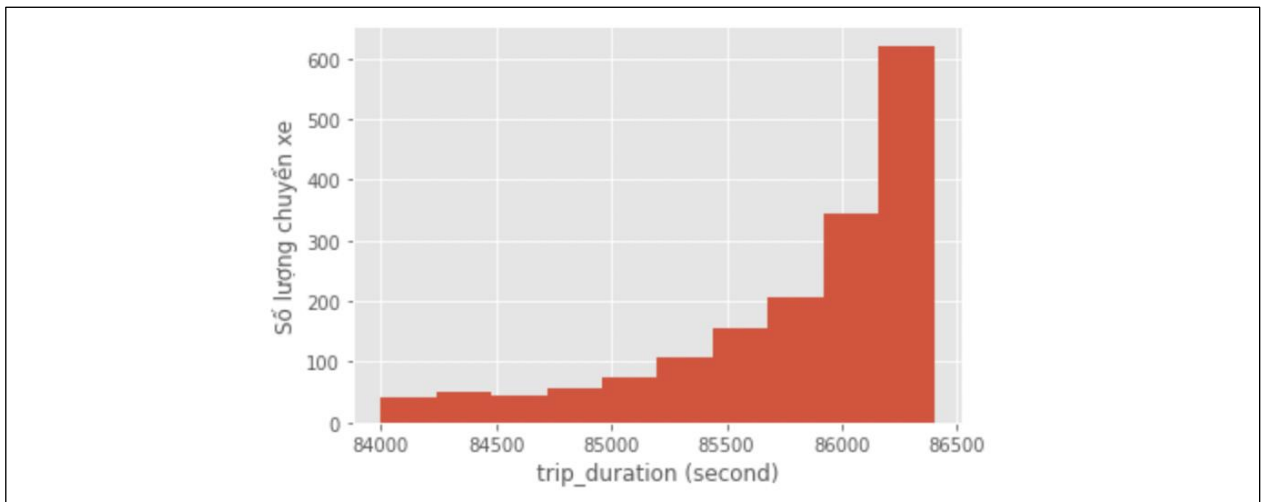
Output[29]:



Output[30]:



Output[31]:



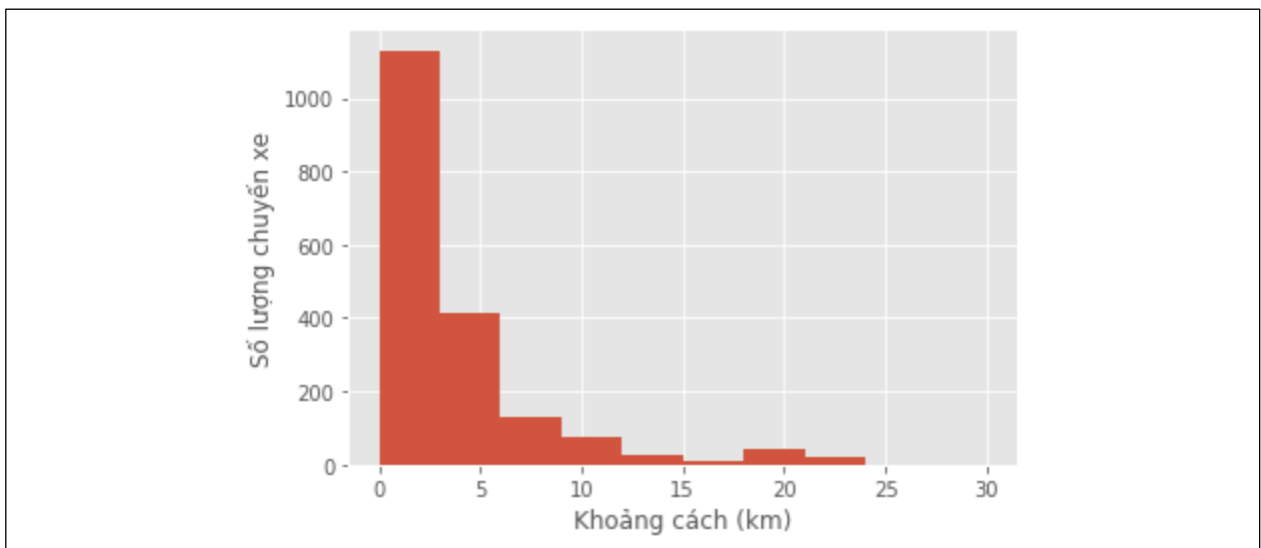
Nhận xét:

- Các chuyến đi trong tập dữ liệu phần lớn có trip\_duration từ 3000 giây trở về trước với số lượng bản ghi rất lớn so với các miền giá trị còn lại. Chỉ riêng với khoảng từ 0 tới 1000 giây đã có tới hơn 1 triệu bản ghi trong khi tập dữ liệu chỉ gần 1 triệu rưỡi dòng.
- Trong khoảng từ 7000 giây đến 86400 giây chúng ta có khá ít bản ghi mặc dù đây là một miền giá trị rất dài, dài hơn rất nhiều so với miền giá trị trước đó nhưng so về số lượng bản ghi thì ít hơn rất nhiều.
- Ta nhận thấy từ cột mốc 82600s trở lên thì số lượng bản ghi có xu hướng tăng. Đây là một khoảng giá trị trip\_duration vô cùng lớn vì nó kéo dài

tới hơn 1 ngày (quá dài so với 1 chuyến taxi thông thường trong một thành phố). Và thông qua các biểu đồ, ta cũng nhận thấy rằng khi thời lượng của trip\_duration càng lớn thì số lượng bản ghi có xu hướng giảm đi. Điều này là hợp lý vì thời lượng của các chuyến taxi thường không dài đến thế và các chuyến đi này chỉ là thiểu số hoặc không tồn tại ở ngoài thực tế. Vì vậy, việc có nhiều chuyến taxi có trip\_duration kéo dài tới xấp xỉ hơn 1 ngày trở lên tăng lên bất ngờ như thế là một bất thường.

Ta cũng sẽ xem tốc độ và quãng đường của các chuyến đi có trip\_duration  $\geq 82600$ .

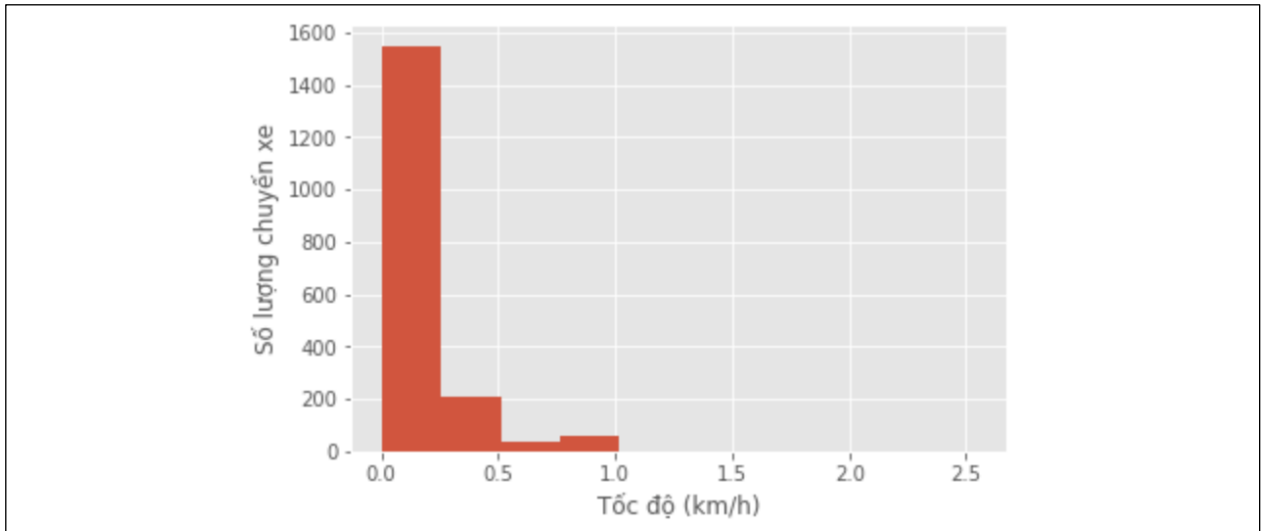
Output[32]:



Ta nhận thấy tuy có thời lượng di chuyển rất dài nhưng các chuyến đi này phần lớn đều di chuyển dưới 10km. Đây là một bất thường khá lớn khi thời gian di chuyển lâu nhưng lại đi được cự ly rất ngắn.



Output[33]:



Ngoài ra các chuyến đi này còn có tốc độ rất chậm khi tất cả đều có tốc độ dưới 1 km/h.

**Quyết định:** Thông qua các khảo sát phía trên, ta sẽ xóa các chuyến đi có thuộc tính `trip_duration` lớn hơn 82600 giây.

Ngoài các chuyến đi quá dài thì chúng ta cũng cần quan tâm đến các chuyến đi có độ dài quá ngắn. Dựa vào thông tin ở phần trước, chúng ta đã biết giá trị bé nhất của cột `trip_duration` là 1 giây nên vì vậy ta sẽ tiến hành đếm số lượng các chuyến có thời lượng 1 giây.

Output[34]:

33

Mở rộng ra với các chuyến có độ dài bé hơn 120 giây:

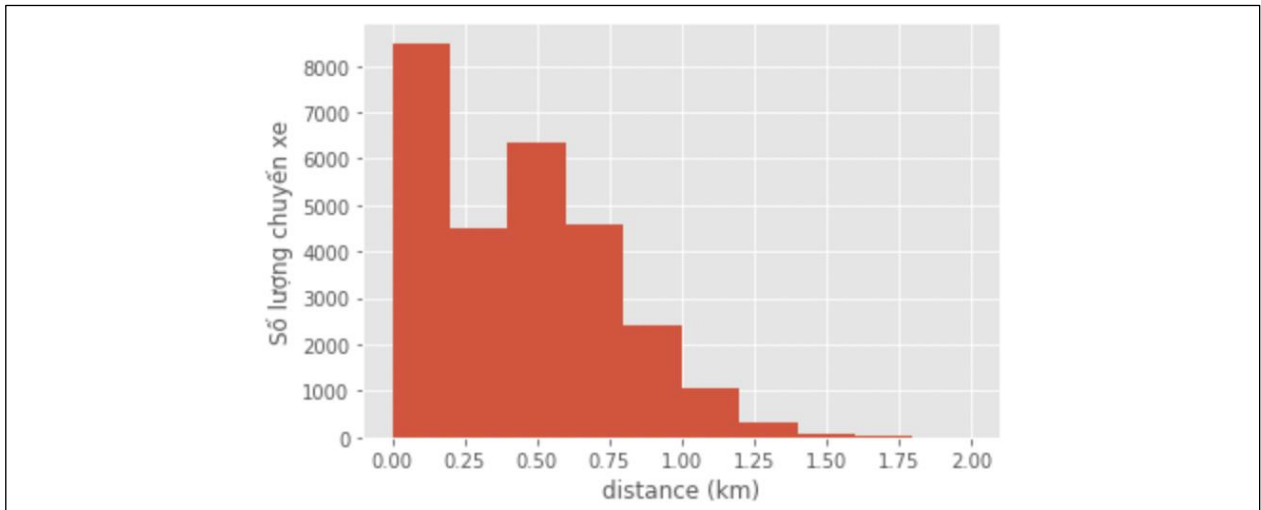
Output[35]:

27817

Ta nhận thấy các chuyến xe dưới 2 phút có số lượng 27817 bản ghi, có thể các chuyến này là do khách hàng hủy chuyến hoặc khách hàng đổi ý khi xe chỉ mới vừa lăn bánh.

Tiếp theo, ta sẽ tiến hành xem thử khoảng cách di chuyển của các chuyến xe có trip\_duration bé hơn 120 giây là bao nhiêu.

Output[36]:



Ta thấy phần lớn các chuyến đi dưới 120 giây đều có khoảng cách di chuyển bé hơn 1km, điều này là hợp lý bởi vì với một khoảng thời gian ngắn như thế thì sẽ không di chuyển được xa. Điều này cho thấy luận điểm khách hàng hủy chuyến hoặc đổi ý trong quá trình di chuyển là có cơ sở tin tưởng được.

Ta sẽ tiếp tục làm việc với các chuyến có quãng đường là 0km:

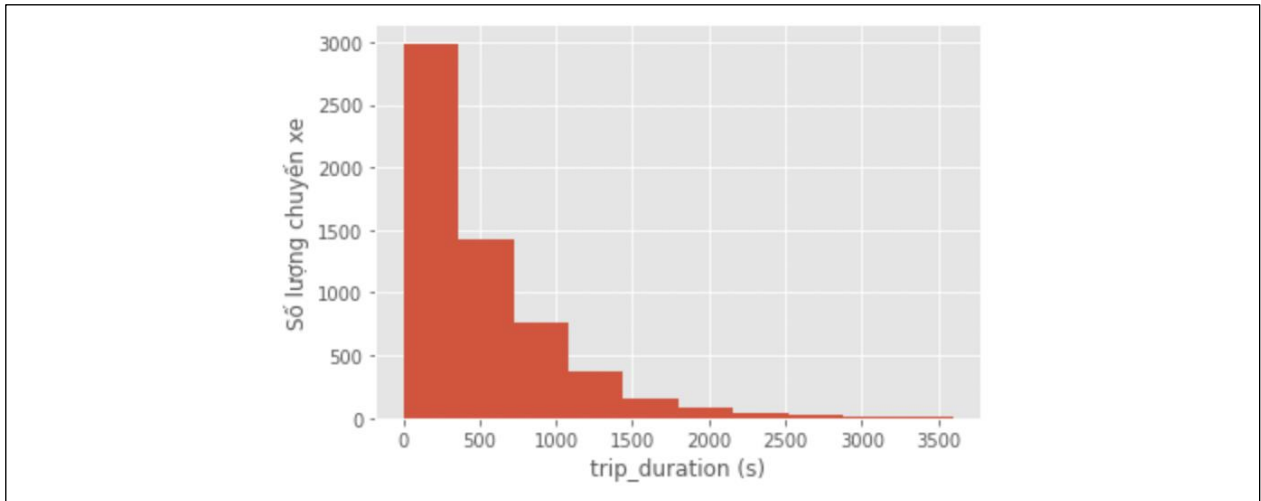
Output[37]:

5897

Chúng ta có 5897 chuyến đi với khoảng cách là 0km.

Ta tiếp tục xem thời lượng di chuyển của các chuyến có độ dài quãng đường bằng 0.

Output[38]:



Output[39]:

1935

Như vậy là chúng ta có 1935 chuyến đi không di chuyển mà có trip\_duration dưới 120 giây. Các chuyến còn lại có trip\_duration trải dài từ 120 giây đến 3600 giây tương ứng với gần 1 giờ.

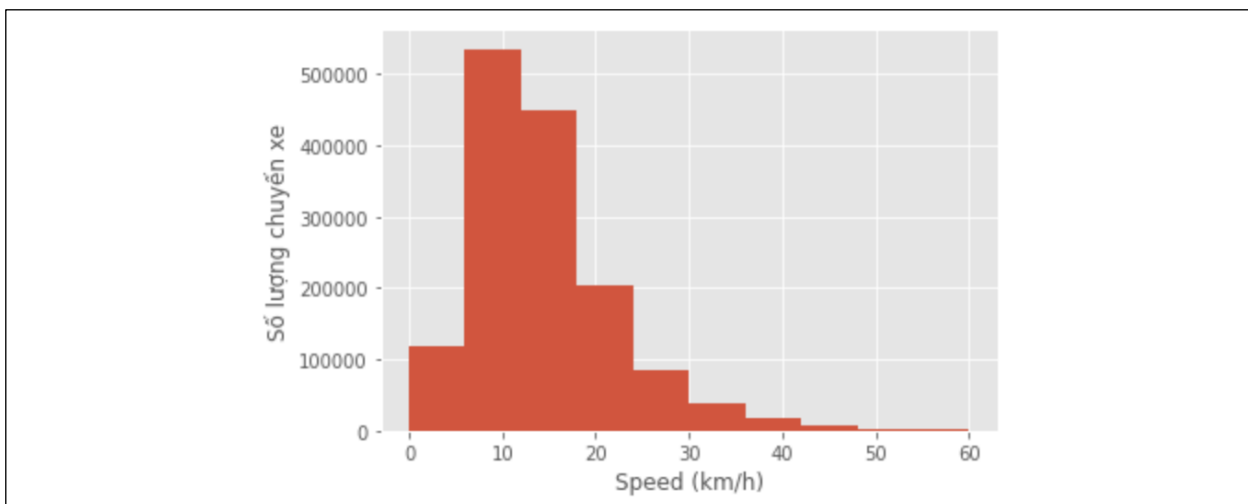
**Quyết định:** Ta sẽ xóa các chuyến xe không di chuyển mà có trip\_duration lớn hơn 120 giây và giữ lại các chuyến xe có trip\_duration bé hơn 120 giây vì có thể các chuyến này bị hủy bởi khách hàng khi xe chưa di chuyển.

#### 2.4.3. Khảo sát cột *speed*.

Output[40]:

```
1176337    9279.581773
910072     6874.214785
184925     5638.682980
974378     5253.665463
377068     4090.789812
1013474    1760.893664
923793     1442.634083
906281     1410.612913
1001028    1408.750365
595540     1355.140122
275644     1292.957998
1107       1266.014753
898154     1243.162682
644165     1137.259158
777670     1094.027657
283102     951.259095
1322903    928.984704
1200843    912.431995
259403     910.453922
467645     835.257183
Name: speed, dtype: float64
```

Ouput[41]:

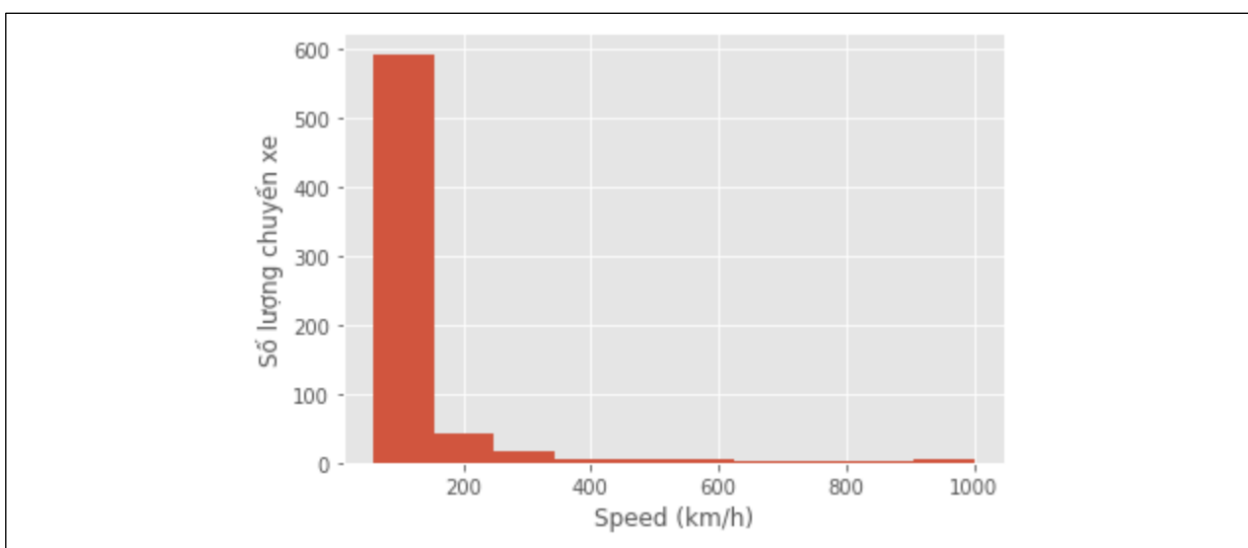


Ouput[42]:

1458469

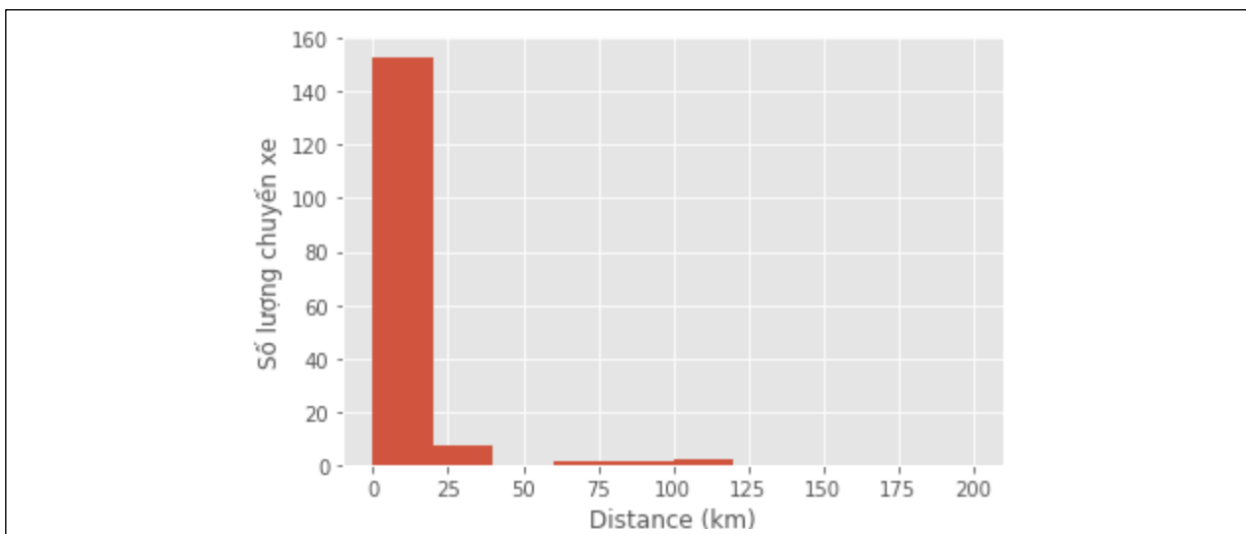
Ta thấy phần lớn các chuyển xe đều có tốc độ di chuyển dưới 100km/h với tổng số 1.458.469 bản ghi. Đây là một tốc độ bình thường đối với các chuyển taxi trong thành phố.

Ouput[43]:

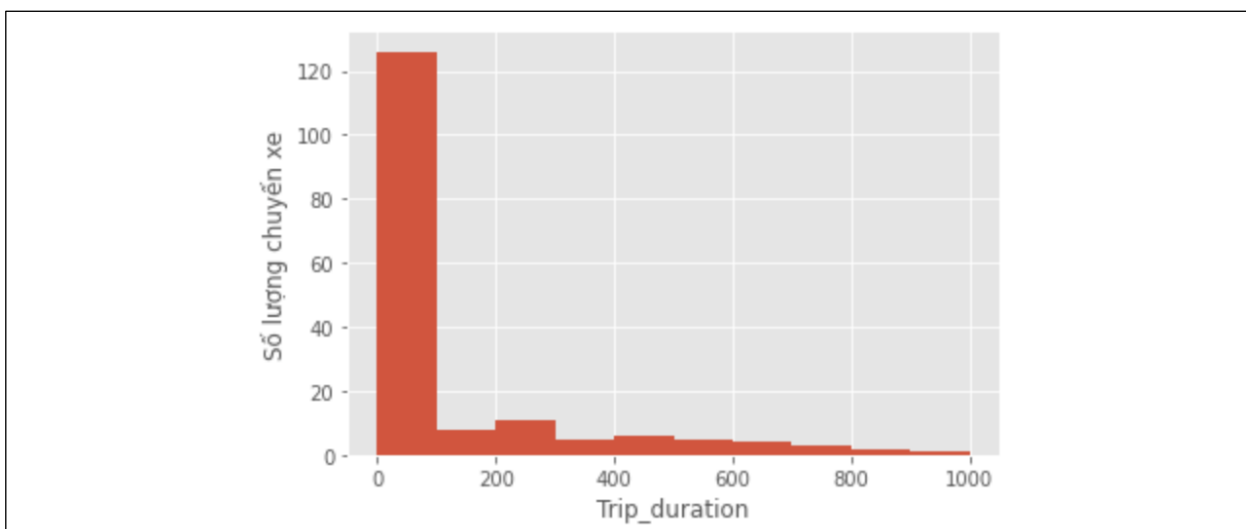


Đối với các chuyển xe có tốc độ di chuyển từ 60km/h trở lên không chiếm số lượng nhiều, trong đó tốc độ từ 60km/h đến 100km/h chiếm phần lớn.

Ouput[44]:



Ouput[45]:



**Quyết định:** Xóa các chuyến đi có tốc độ di chuyển lớn hơn 100km/h.

## 2.5. Tiền xử lý dữ liệu (tiếp tục).

2.5.1. Xóa các bản ghi có *passenger\_count* = 0, 7, 8, 9.

2.5.2. Xóa các chuyến có *trip\_duration* lớn hơn 82600.

2.5.3. Xóa các chuyến có *distance* = 0 và *trip\_duration* lớn hơn 120 giây.

2.5.4. Xóa các chuyến có tốc độ di chuyển từ 100km/h trở lên.

## 2.6. Khám phá dữ liệu (tiếp tục).

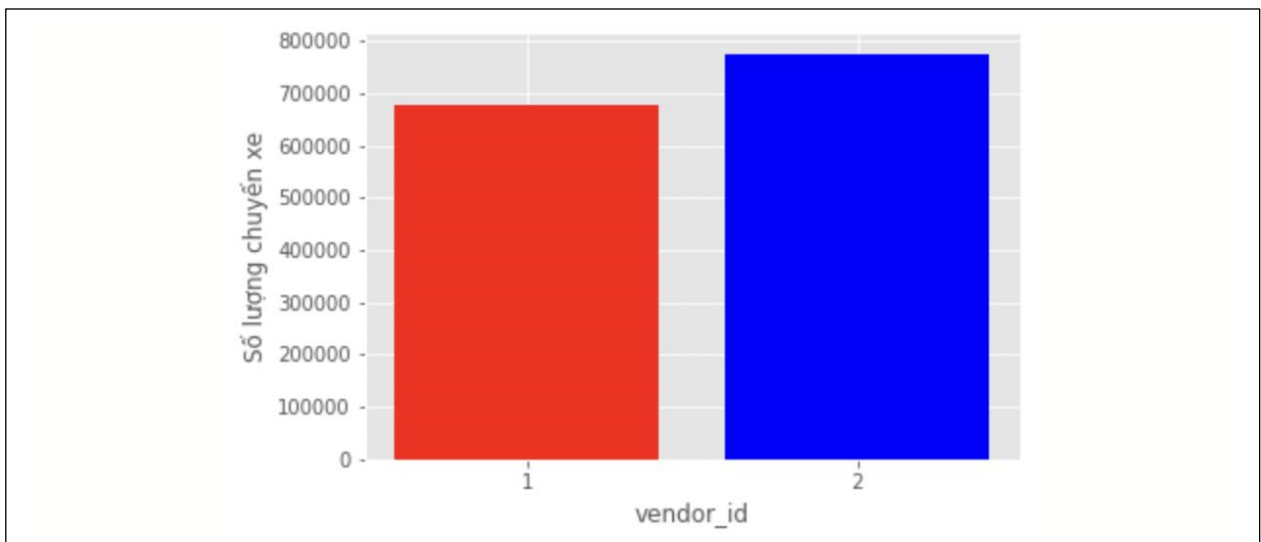
Với mỗi cột có kiểu dữ liệu dạng categorical, các giá trị được phân bố như thế nào?

vendor\_id và store\_and\_flag là 2 cột có kiểu dữ liệu dạng categorical.

### 2.6.1. Cột *vendor\_id*:

Cột vendor\_id cho biết nhà cung cấp được liên kết với hồ sơ chuyển đi.

Output[46]:



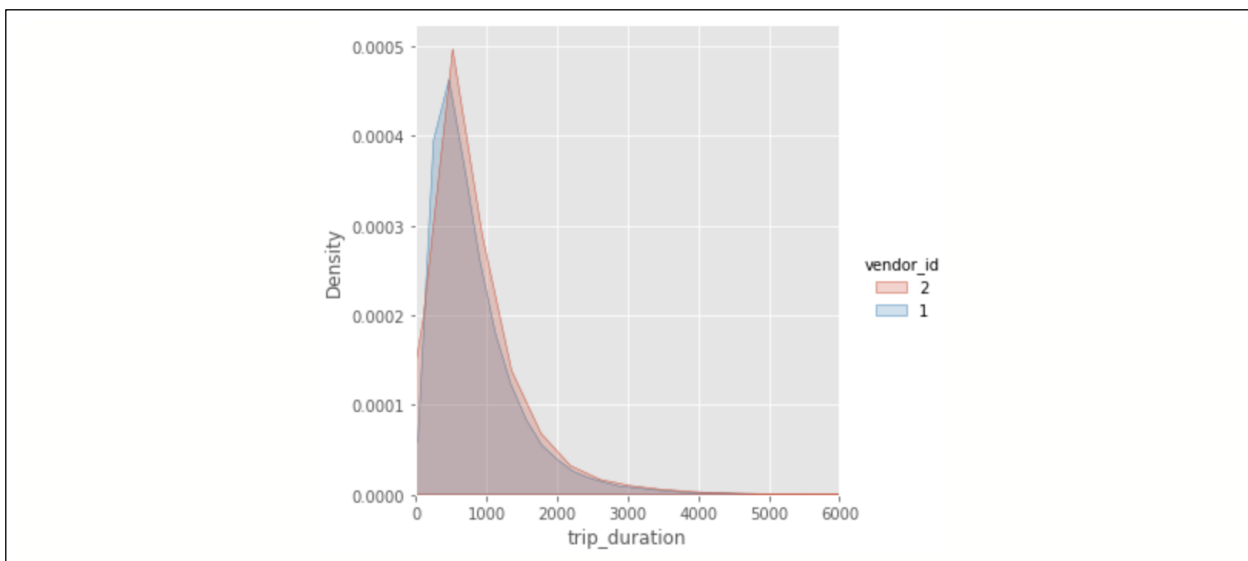
Dựa vào biểu đồ trên ta thấy Vendor 2 cung cấp nhiều hơn Vendor 1 khoảng 100000 bản ghi.

Output[47]:

```
1    677432
2    775168
Name: vendor_id, dtype: int64
```

So sánh sự khác biệt giữa Vendor\_id 1 và Vendor\_id 2.

Output[48]:



Nhận thấy sự khác biệt giữa Vendor\_id 1 và Vendor\_id 2 là không đáng kể.

#### 2.6.2. *Cột store\_and\_fwd\_flag:*

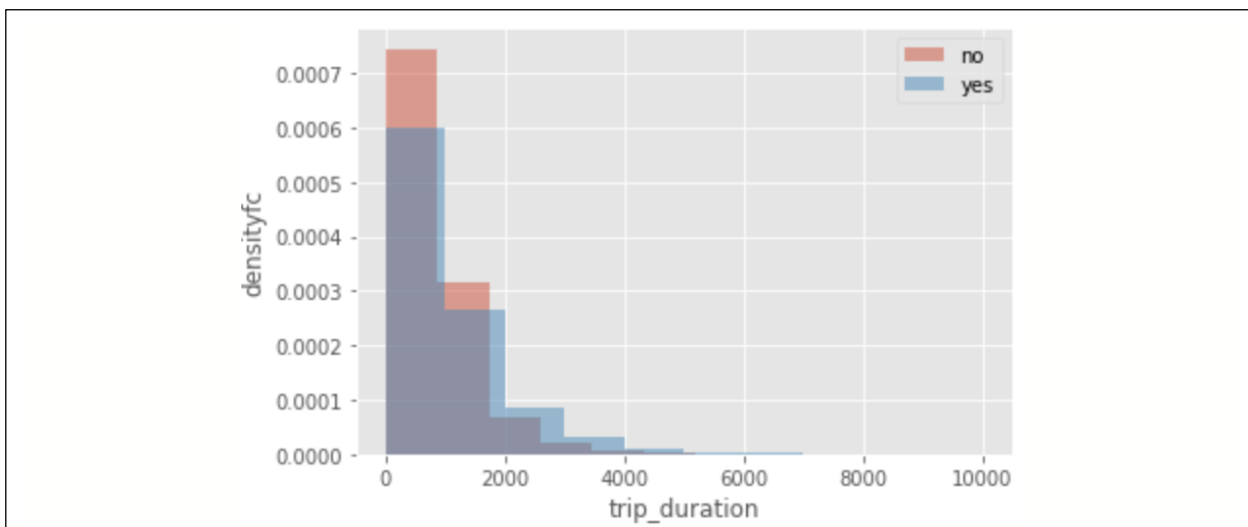
Output[49]:

0.5519069255128735

Khoảng 0.55% các chuyến có bản ghi được ghi lại trong bộ nhớ của phương tiện trước khi được gửi đến cho nhà cung cấp.

Ta tiếp tục tìm hiểu mối liên hệ của nó với trip\_duration.

Output[50]:



Dựa vào biểu đồ trên, ta nhận thấy sự khác biệt không đáng kể. Vì vậy sẽ không khám phá thêm về thuộc tính này.

### 3. PHÂN TÍCH DỮ LIỆU.

Mục tiêu của bài toán là `trip_duration` nên nhóm mình đặt ra các câu hỏi về ảnh hưởng của thời gian và địa điểm của điểm đón/trả liệu có ảnh hưởng đến `trip_duration`. Trên thực tế, thời gian và địa điểm có ảnh hưởng đến độ dài chuyến đi, có thể vào giờ cao điểm ở một số khu vực sẽ bị di chuyển chậm hơn bình thường. Để trả lời cho các giả thuyết này, chúng ta sẽ đi qua 2 câu hỏi lớn.

#### 3.1. Câu hỏi phân tích số 1: `trip_duration` thay đổi như thế nào theo các ngày trong tuần và các giờ trong ngày?

Tiếp theo, ta sẽ tiếp cận với 3 khung giờ như sau:

- Giờ cao điểm: Là khung giờ có số lượng pickup cao.
- Giờ bình thường: Là khung giờ có số lượt pickup trung bình (nằm ở khoảng giữa).
- Giờ thấp điểm: Là khung giờ có số lượt pickup thấp.

Tiếp theo ta sẽ trích xuất thông tin có trong cột `pickup_datetime` thành các thông tin riêng lẻ như: Tháng, giờ đón khách, ngày hôm đó thuộc thứ mấy trong tuần, là tuần thứ bao nhiêu trong năm?

Khi làm việc với các đối tượng `datetime` của `pandas` ta có thể sử dụng trình truy cập `.dt` để truy cập các thuộc tính khác nhau từ các đối tượng này. Điều này có nghĩa là chúng ta có thể trích xuất các phần khác nhau từ một đối tượng ngày giờ, chẳng hạn như tháng, ngày, v.v

Output[51]:

	Month	Weekday	Hour	Week	Distance	Trip_duration	Speed
0	3	0	17	11	1.502172	455	11.885316
1	6	6	0	23	1.808660	663	9.820778
2	1	1	11	3	6.379687	2124	10.813029
3	4	2	19	14	1.483632	429	12.450063
4	3	5	13	12	1.187038	435	9.823760

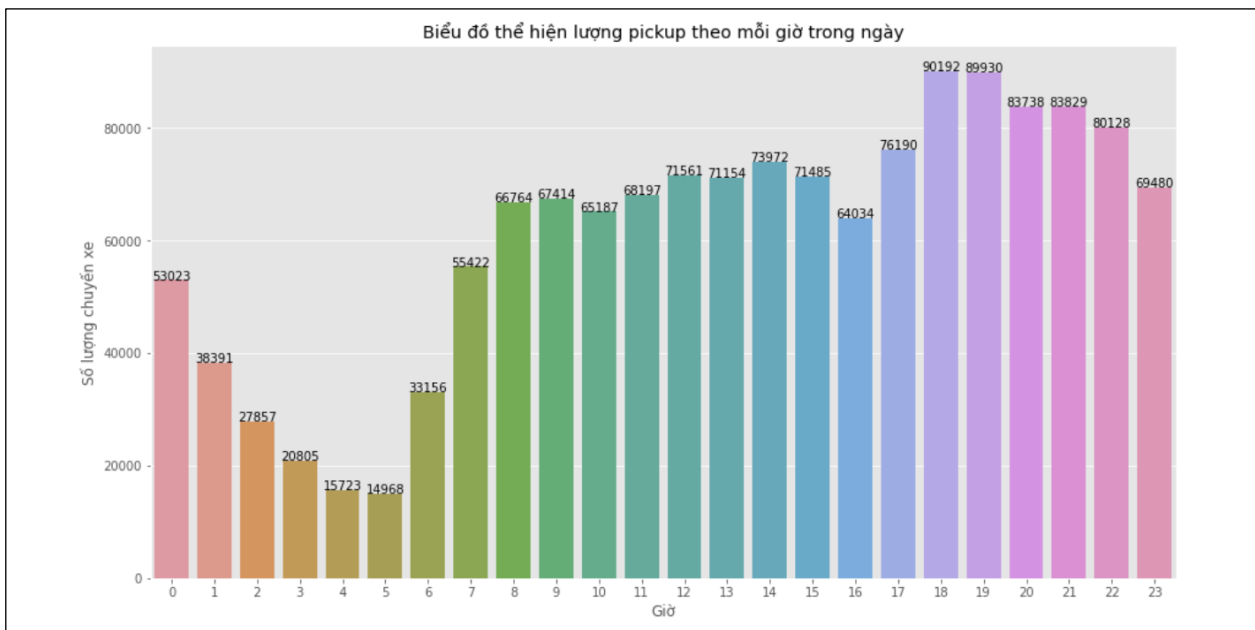


Trong đó:

- Month: tương ứng với tháng.
- Weekday: Thứ trong tuần tương ứng với: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday được gán nhãn tương ứng từ 0 tới 6.
- Hour: Giờ đón khách.
- Week: Thứ tự tuần trong năm.
- distance: Khoảng cách di chuyển của chuyến đi.
- trip\_duration: Thời gian di chuyển của chuyến xe.
- speed: Tốc độ di chuyển của chuyến đi.

Tiến hành tìm kiếm khung giờ nào là cao điểm, trung bình, thấp điểm.

Output[52]:



Dựa vào biểu đồ trên, ta có thể đưa ra nhận xét rằng:

- Giờ cao điểm nằm trong khoảng 18-23h.
- Giờ bình thường nằm trong khoảng 8-17h.
- Giờ thấp điểm nằm trong khoảng 0-7h.

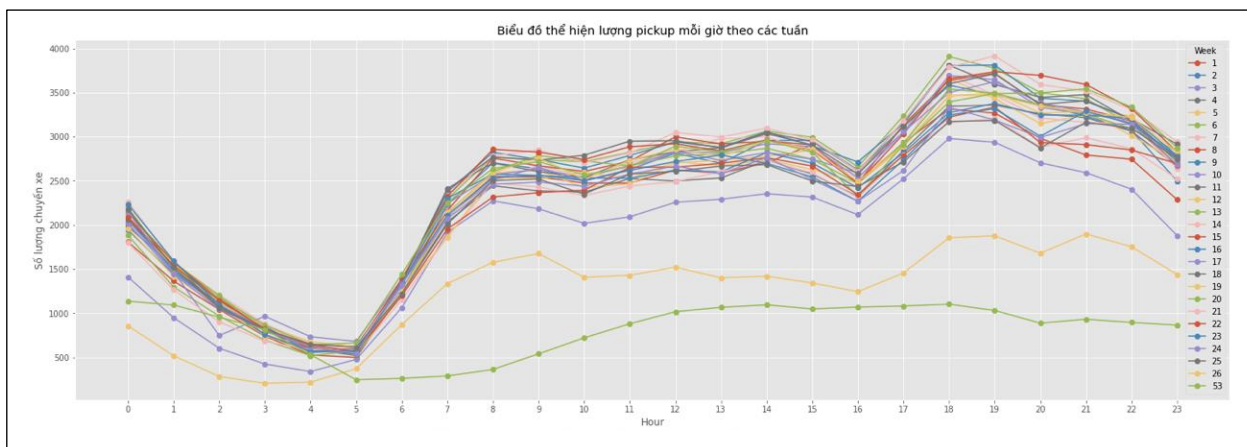
Để có cái nhìn rộng hơn, ta tiếp tục vẽ số lượng pickup theo giờ biểu diễn theo từng tuần và từng tháng.

Để biết được số lượng pick\_up theo từng giờ ở từng tuần khác nhau ta sẽ tiến hành nhóm các dòng dữ liệu thành các nhóm theo từng giờ và từng tuần thông qua hàm groupby() có sẵn trong thư viện pandas khi làm việc với các dataframe, đồng thời ta cũng sử dụng size() để đếm số dòng trong mỗi nhóm này (hàm size() sẽ đếm luôn những dòng có giá trị là NA). Kết quả trả về sẽ là 1 series.

Và để vẽ được đồ thị biểu diễn số lượng pickup theo từng giờ và từng tuần, tiếp theo ta sẽ cần sử dụng hàm unstack() để biểu diễn kết quả đã groupby từ phía trên thành 1 DataFrame. Hàm này sẽ hỗ trợ xoay các nhãn chỉ mục (phải được phân cấp) 1 cấp. Tham khảo thêm [tại đây](#).

### Biểu đồ thể hiện lượng pickup mỗi giờ theo các tuần:

Output[53]:



Ta nhận thấy ở tuần thứ 53 và 26 có sự bất thường so với các tuần còn lại khi có biểu đồ đường bị lệch về phía dưới, nguyên do là số lượng bản ghi ở 2 tuần này ít hơn các tuần khác rất nhiều đã dẫn đến tình trạng như trên.

Output[54]:

```
53    19931
26    29640
3     45101
21    52099
1     52965
Name: Week, dtype: Int64
```

Output[55]:

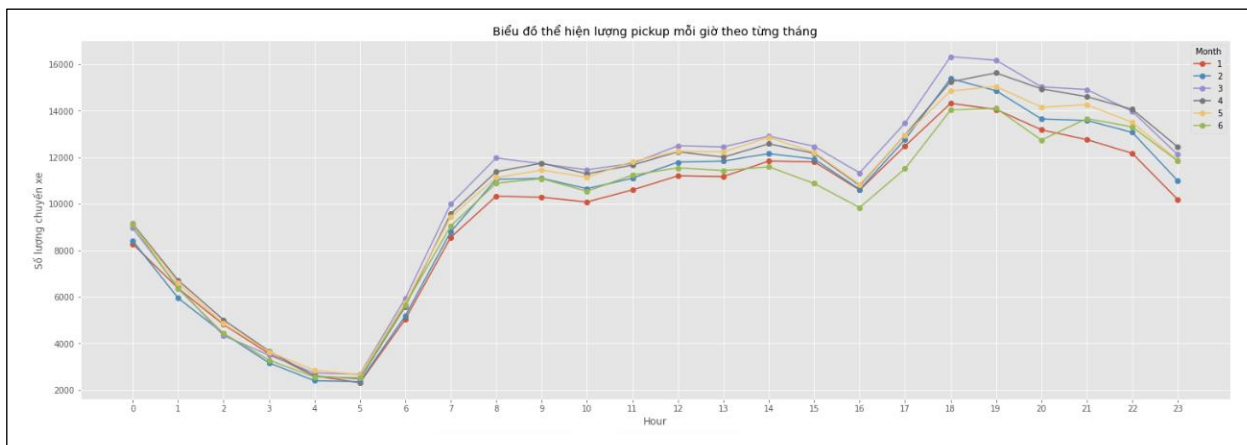
Weekday	0	1	2	3	4	5	6
Week							
1	6687.0	7165.0	7336.0	7621.0	8199.0	8534.0	7423.0
2	7248.0	7769.0	8327.0	8238.0	8513.0	8759.0	7844.0
3	7111.0	7906.0	7989.0	8336.0	8755.0	1640.0	3364.0
4	6057.0	6905.0	7541.0	8014.0	8697.0	9135.0	7640.0
5	7035.0	7631.0	7790.0	8345.0	8452.0	8857.0	7795.0
6	7087.0	7591.0	8288.0	9083.0	9075.0	9267.0	8439.0
7	7335.0	7622.0	8161.0	8542.0	8742.0	8883.0	7653.0
8	7171.0	8131.0	8416.0	8610.0	9023.0	9193.0	7739.0
9	7369.0	7810.0	8547.0	8855.0	9189.0	9560.0	8098.0
10	7372.0	7843.0	8097.0	8516.0	8847.0	9204.0	7709.0
11	8062.0	7997.0	8331.0	8227.0	9075.0	9095.0	8156.0
12	7235.0	7600.0	7829.0	8496.0	8312.0	8495.0	7333.0
13	7210.0	7916.0	8004.0	8165.0	8608.0	9283.0	8048.0
14	7869.0	8360.0	8435.0	8632.0	9177.0	9756.0	8345.0
15	7458.0	8266.0	8252.0	8819.0	9222.0	9568.0	8220.0
16	7431.0	7962.0	8311.0	8057.0	8034.0	8307.0	7491.0
17	7013.0	7829.0	7922.0	8515.0	8696.0	8741.0	7912.0
18	7324.0	8427.0	8500.0	8866.0	9161.0	9252.0	7771.0
19	7140.0	7756.0	7911.0	8483.0	8449.0	9101.0	8515.0
20	8054.0	8200.0	7939.0	8422.0	8772.0	9090.0	7718.0
21	7281.0	7725.0	8169.0	8099.0	7507.0	6972.0	6346.0
22	5551.0	7101.0	7901.0	8283.0	8299.0	8850.0	7555.0
23	7373.0	7817.0	7981.0	8302.0	8342.0	8122.0	7093.0
24	7078.0	7627.0	7979.0	8515.0	8103.0	7627.0	6931.0
25	6890.0	7751.0	7772.0	8078.0	8193.0	8108.0	7055.0
26	7288.0	7209.0	7591.0	7552.0	NaN	NaN	NaN
53	NaN	NaN	NaN	NaN	7129.0	6487.0	6315.0

- Ở tuần thứ 26: Dữ liệu bị khuyết ở các ngày thứ sáu, thứ bảy và chủ nhật.
- Ở tuần thứ 53: Dữ liệu bị khuyết ở các ngày thứ hai, thứ ba, thứ tư và thứ năm.

Do đó từ giờ, khi vẽ biểu đồ thể hiện các đặc điểm của dữ liệu liên quan đến từng tuần, để đảm bảo độ nhất quán cũng như tính liên tục của các nét vẽ trong biểu đồ, ta sẽ tạm thời bỏ qua 2 tuần này.

### **Biểu đồ thể hiện lượng pickup mỗi giờ theo từng tháng:**

Output[56]:



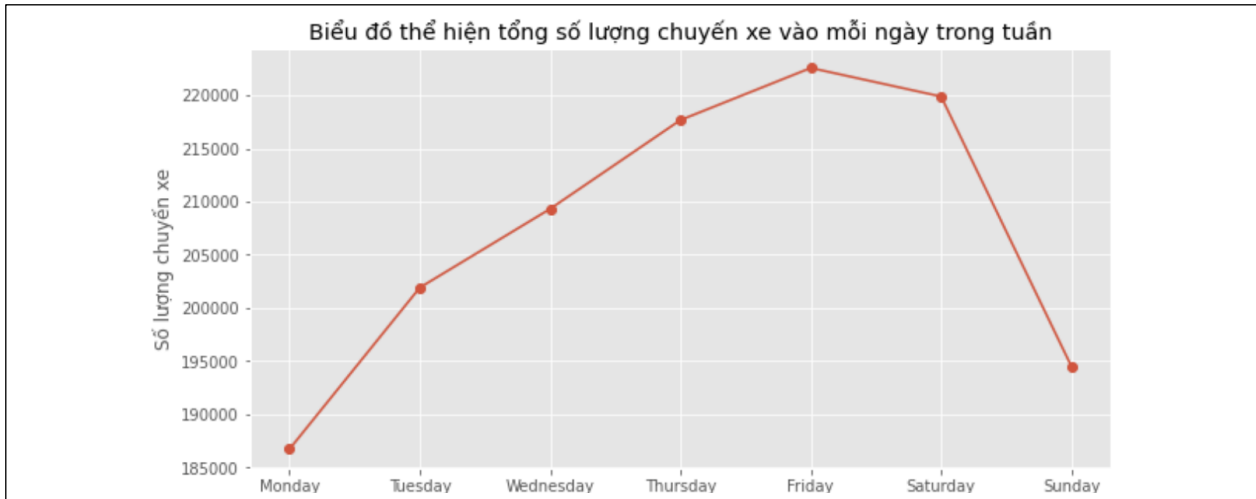
Dựa vào 2 đồ thị trên ta có thể thấy các tháng và các tuần trong dữ liệu gần như tương tự nhau.

Dựa vào các kết quả phân tích ở trên ta có thể kết luận như sau:

- Giờ cao điểm nằm trong khoảng 18-23h.
- Giờ bình thường nằm trong khoảng 8-17h.
- Giờ thấp điểm nằm trong khoảng 0-7h.

Ta tìm hiểu xem số lượng pickup thay đổi như thế nào theo các ngày trong tuần.

Output[57]:

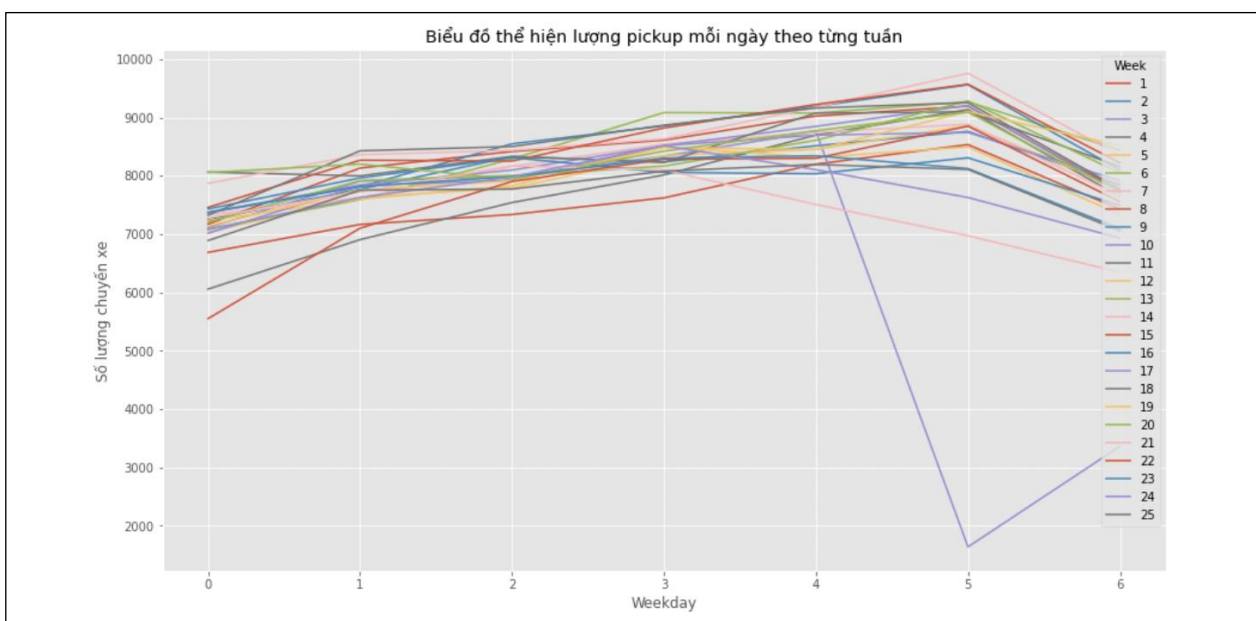


Dựa vào đồ thị trên, ta có thể đưa ra giả thuyết rằng: Ngày thứ 2 chính là ngày có số lượng pickup thấp nhất trong tuần, xếp sau chính là ngày chủ nhật. Ngày thứ 6 và thứ 7 là 2 ngày có số lượng pickup cao nhất trong các ngày trong tuần. Và nhìn chung, số lượng pickup có xu hướng tăng dần từ đầu tuần và đạt đỉnh điểm vào ngày thứ 7, sau đó bắt đầu hạ xuống thấp vào ngày chủ nhật.

Để có cái nhìn rộng hơn, ta sẽ tiếp tục xem lượng pickup mỗi ngày theo từng tuần để đưa ra một kết luận cho giả thuyết trên.

**Biểu đồ thể hiện lượng pickup mỗi ngày theo từng tuần:**

Output[58]:

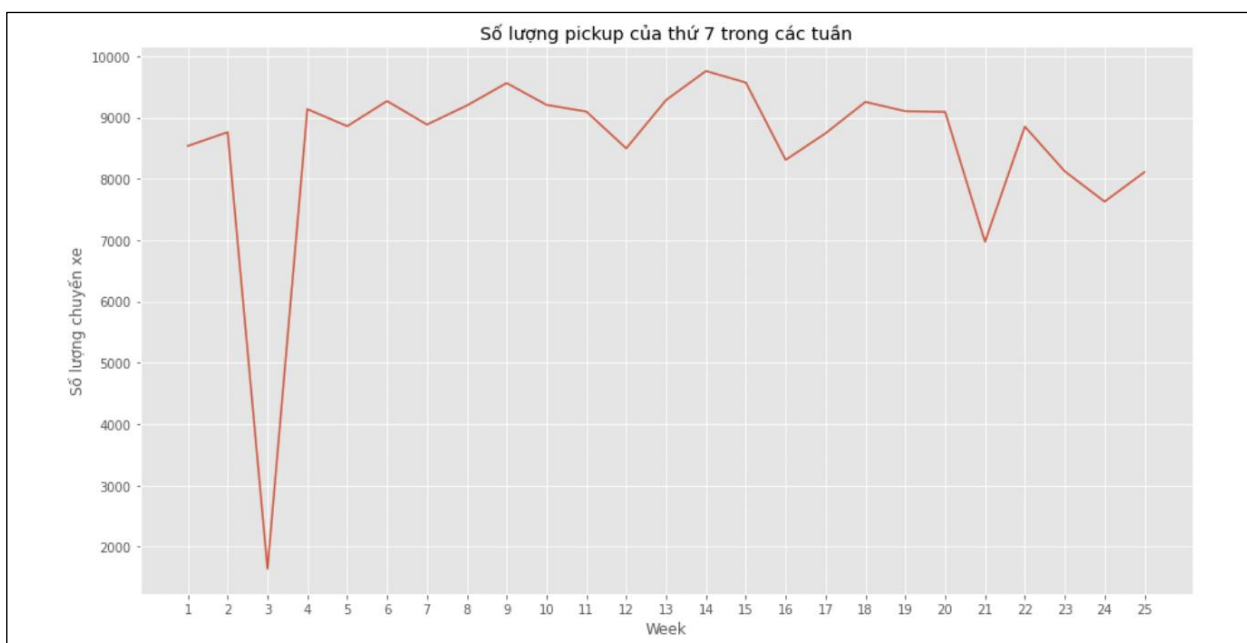


Dựa vào đồ thị trên thì ta đã có một cái nhìn tổng quát hơn về số lượng pickup thay đổi theo ngày qua từng tuần, và từ đây ta có thể củng cố được giả thuyết mà ta đã đặt ra phần trước, đó là: Lượng pickup tăng dần từ ngày đầu tuần và sẽ đạt đỉnh vào thứ 6 hoặc thứ 7, sau đó sẽ giảm xuống khi bước qua ngày chủ nhật.

Đồng thời, ta cũng nhận thấy vào ngày thứ 7 của một tuần nào đó đã có lượng pickup thấp bất thường. Ta sẽ tiến hành tìm hiểu.

Đầu tiên, ta sẽ vẽ biểu đồ thể hiện số lượng pickup vào các ngày thứ 7 của từng tuần cụ thể:

Output[59]:



Dựa vào đồ thị, ta đã xác định được đây là thứ 7 của tuần thứ 3. Tiếp theo, ta sẽ xác định xem đây là ngày bao nhiêu.

Output[60]:

	Month	Weekday	Hour	Week	Distance	Trip_duration	Speed
308	1	5	8	3	1.000119	1197	3.007878

Output[61]:

```
pickup_datetime    2016-01-23 08:46:42
dropoff_datetime    2016-01-23 09:06:39
Name: 308, dtype: object
```

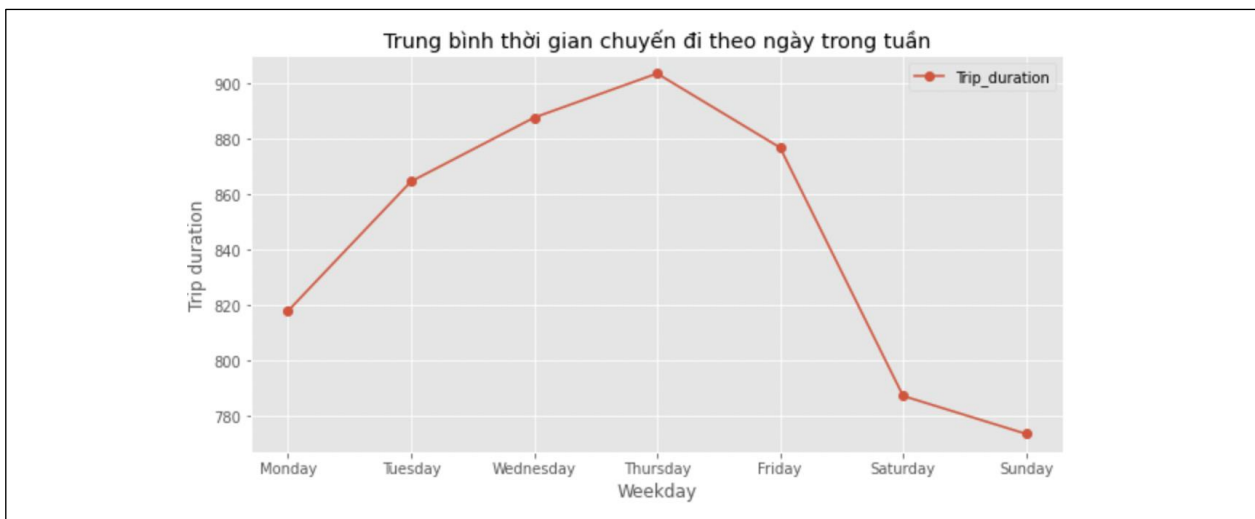
Qua kết quả phía trên, ta thấy đây là ngày 23/01/2016 và là thời điểm trùng với một trận bão tuyết rất lớn tại thành phố New York, theo nguồn thông tin ghi nhận tại [bài báo này](#). Do đó lượng pickup của chúng ta vào ngày này rất thấp, chỉ với 1640 bản ghi.

Ta đi đến kết luận:

- Số lượng pickup theo chu kỳ như sau: pickup ở mức thấp vào cuối tuần (chủ nhật) và đầu tuần (thứ 2), sau đó tăng dần lên đỉnh điểm vào thứ 6 hoặc 7 và giảm xuống vào chủ nhật.
- Số lượng pickup phụ thuộc vào thời tiết, khi mà có sự kiện thời tiết cực đoan thì số lượng pickup sẽ giảm rất rõ rệt.

Đánh giá về trip\_duration đối với mỗi khung giờ và theo các ngày trong tuần:

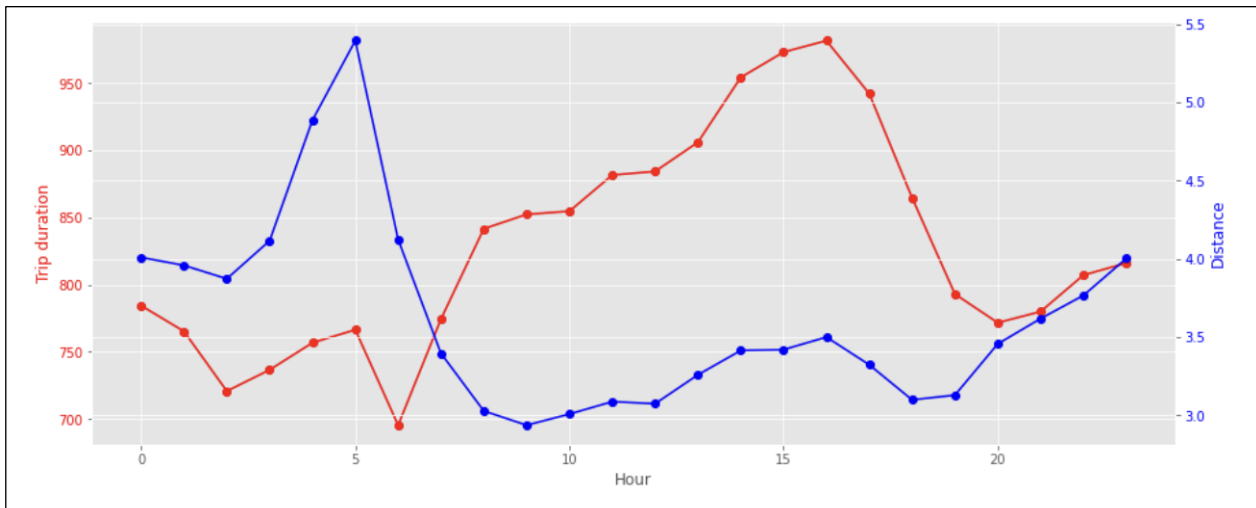
Output[62]:



Dựa vào đồ thị, thời gian trung bình chuyến đi ở mức cao ở giữa tuần và thấp hơn đáng kể vào ngày thứ bảy và chủ nhật. Ta có thể đưa ra giả thuyết là cuối tuần, ít người đi làm sẽ khiến cho các chuyến đi nhanh hơn, hoặc vào ngày cuối tuần các chuyến đi sẽ ngắn hơn, tiết kiệm thời gian hơn.

Ta sẽ tìm hiểu vận tốc giữa các khung giờ như thế nào.

Output[63]:



Ta đưa ra một nhận xét là thường trong khung giờ cao điểm, số lượng pickup nhiều thì thời gian di chuyển sẽ lâu hơn, nhưng ở đây biểu đồ lại cho thấy thời gian di chuyển (trip\_duration) của khung giờ cao điểm lại nhanh hơn giờ bình thường và tiệm cận với khung giờ thấp điểm.

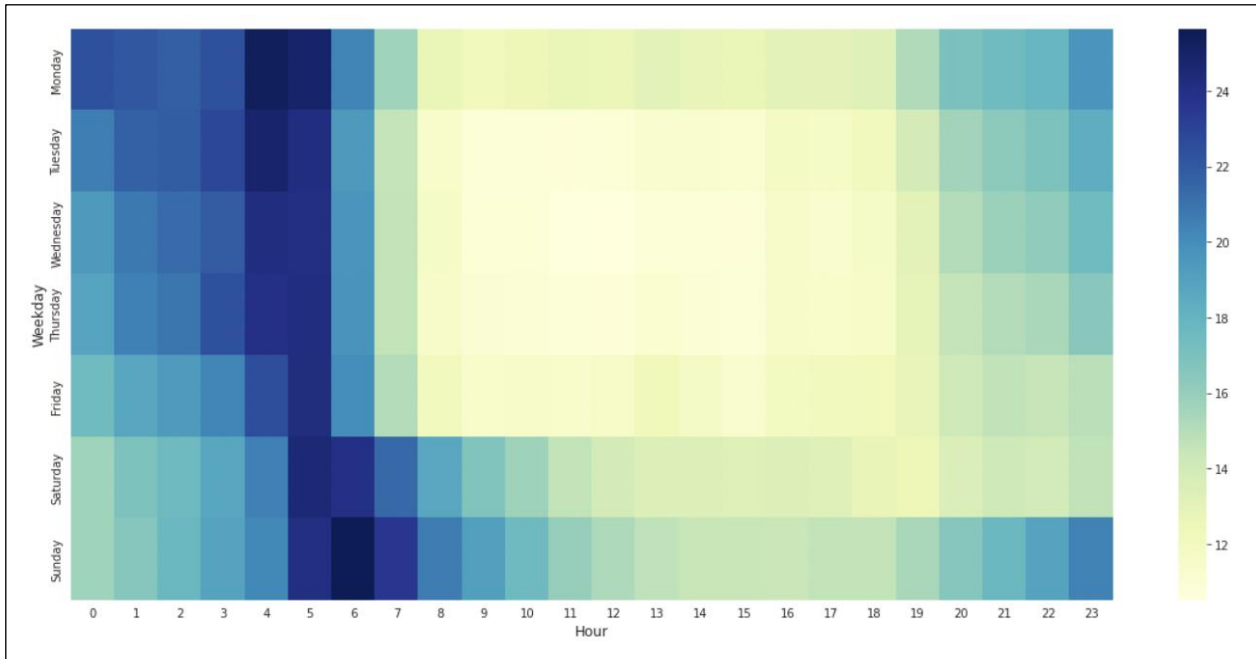
Ta có thể thấy:

- Ở khung giờ cao điểm 18-23h: Khoảng cách di chuyển xa hơn khung giờ bình thường nhưng lại có thời gian di chuyển nhanh hơn => Có thể do mật độ giao thông thấp, di chuyển nhanh.
- Ở khung giờ bình thường 8-17h: Khoảng cách di chuyển gần nhưng thời gian di chuyển di chuyển rất chậm. => Mật độ giao thông rất cao, có thể là kẹt xe.
- Ở khung giờ thấp điểm 0-7h: Khoảng cách di chuyển cao nhất, thời gian di chuyển cũng nhanh nhất => Có thể do mật độ giao thông thưa thớt, đường vắng vẻ.

Để củng cố thêm giả thuyết thì chúng ta sẽ xem Heatmap của vận tốc.



Output[64]:



- Chúng ta có thể thấy trong khung giờ từ 0h-7h thì vận tốc trung bình sẽ cao hơn các khung giờ khác.
- Ở các ngày từ thứ 2 đến thứ 6 vào các khung giờ 7h-19h thì tốc độ trung bình của các chuyến đi tương đối thấp.
- Vào 2 ngày cuối tuần là thứ 7 và chủ nhật, từ sau 11h tốc độ trung bình của chuyến đi cũng bắt đầu giảm xuống.

**Quyết định:** Thêm 1 feature mới là "Low\_Speed\_Time" vào dataset, thể hiện các khung giờ có tốc độ di chuyển thấp, với giá trị là 1 nếu khung giờ là từ 8h-18h từ thứ 2 đến thứ 6, khung giờ 11h-22h đối với thứ 7 và 11h-18h đối với chủ nhật. Ngoài các khung giờ trên sẽ là giá trị 0.

Output[65]:

store_and_fwd_flag	trip_duration	distance	speed	Low_speed_time
False	455	1.502172	11.885316	1
False	663	1.808660	9.820778	0
False	2124	6.379687	10.813029	1
False	429	1.483632	12.450063	0
False	435	1.187038	9.823760	1

Dựa vào các phân tích đã tiến hành ở phía trên, ta có thể đưa ra 1 số kết luận liên quan đến câu hỏi số 1 như sau:

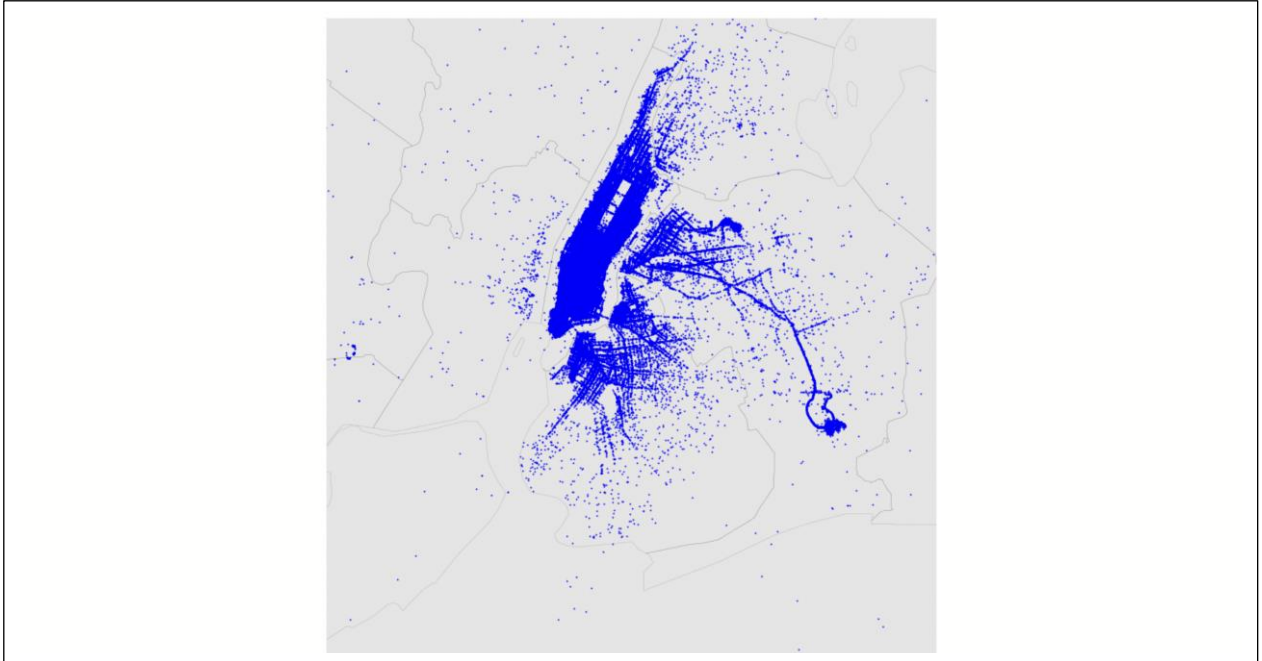
- Số lượng đặt chuyến vào các ngày thứ 2 và chủ nhật có phần thấp hơn so với các ngày còn lại trong tuần. Trong khi đó các ngày giữa tuần như thứ 6, thứ 7 được nhiều lượt pickup hơn so với các ngày còn lại.
- Những ngày có hiện tượng thời tiết cực đoan sẽ có lượng pickup rất thấp, do người dân có xu hướng không đi ra ngoài vào các ngày này. Vậy ta có kết luận rằng lượng pickup bị ảnh hưởng đáng kể bởi thời tiết, nhất là các hình thái thời tiết cực đoan.
- Các chuyến đi trong khung giờ từ 0h - 7h là những chuyến đi dài với tốc độ nhanh nhất so với khung giờ khác, có thể vì đây là thời điểm có mật độ giao thông thông thoáng.
- Các chuyến đi trong khung giờ từ 8h - 17h là những chuyến đi có khoảng cách di chuyển ngắn nhưng lại tốc độ chậm nhất trong các khung giờ.
- Các chuyến đi ở khung giờ từ 18h - 23h là những chuyến đi nằm trong khung giờ cao điểm, có khoảng cách di chuyển ngắn với tốc độ nhanh.

### **3.2. Câu hỏi phân tích số 2: Có gì đáng chú ý ở vị trí điểm đón và trả khách của các chuyến đi hay không?**

Ở phần này, ta sẽ tìm hiểu xem liệu các điểm đón trả khách trong tập dữ liệu có đem đến thông tin hữu ích gì cho chúng ta hay không, ví dụ như nơi nào tập trung nhiều chuyến đi nhất hay liệu vị trí đón trả khách cũng có thể làm ảnh hưởng đến độ dài của chuyến đi hay không?

Tại đây, chúng ta sẽ sử dụng thư viện matplotlib basemap toolkit để vẽ các bản đồ minh họa, ưu điểm của thư viện này là rất nhẹ và dễ dàng để làm quen, sử dụng.

Output[66]:



Dựa vào biểu đồ trên, ta thấy:

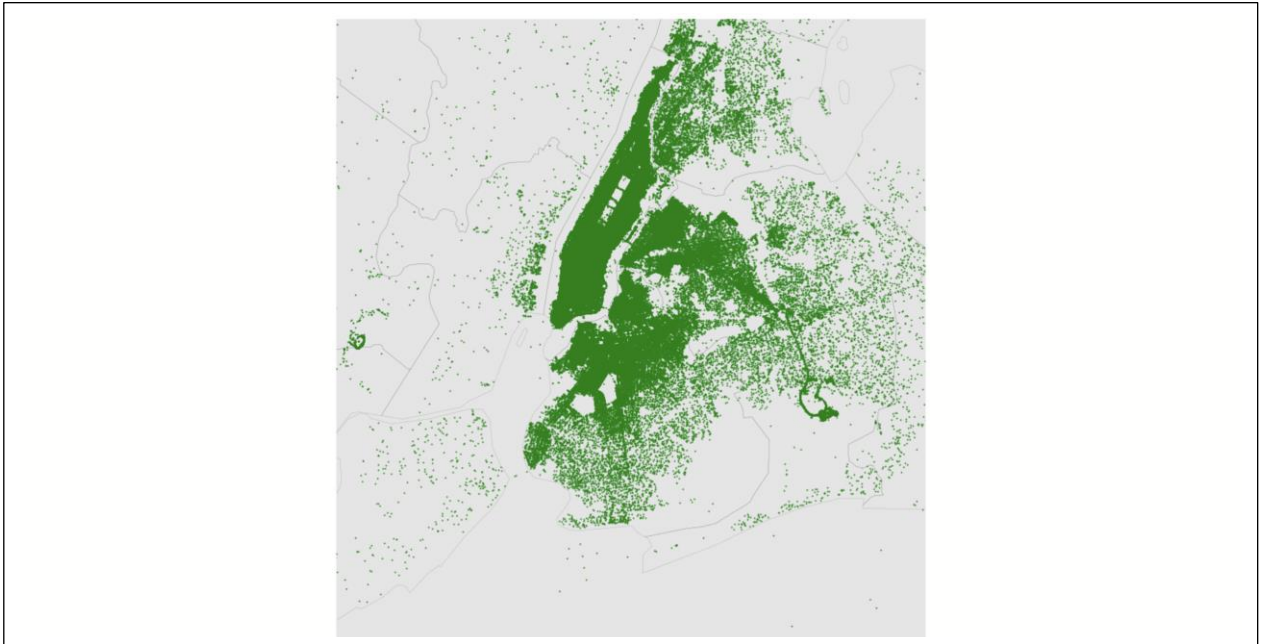
- Các điểm đón khách đa số đều nằm ở quận Manhattan. Đây là quận có dân số đông nhất NYC, có nhiều công viên, nhà hàng, địa điểm giải trí nên vì vậy khách du lịch có lẽ thường di chuyển từ địa điểm này đến địa điểm khác trong khu vực bằng taxi.
- Ta cũng nhận thấy có một con đường nối liền quận Manhattan với một địa điểm nhỏ ở góc dưới phía bên tay phải cũng có khá nhiều chuyến xe đón khách ở đây - Trên thực tế, đây chính là Sân bay quốc tế John F. Kennedy, sân bay này cũng nằm khá xa trung tâm quận Manhattan.
- Ngoài ra cũng có một sân bay khác có vị trí gần trung tâm hơn so với sân bay John F. Kennedy, đó là Sân bay LaGuardia cũng là nơi tập trung nhiều lượt đến và đi trong ngày.

Nhận xét chủ quan:

- Chắc hẳn sẽ có một số chuyến đi đường dài đến và đi từ sân bay.
- Tương tự như vậy, thời gian trung bình cho các chuyến xe được đón đến hoặc đến sân bay sẽ dài hơn các chuyến xe đi và đến trong trung tâm.

## Biểu đồ thể hiện sự phân bố của các điểm trả khách (dropoff):

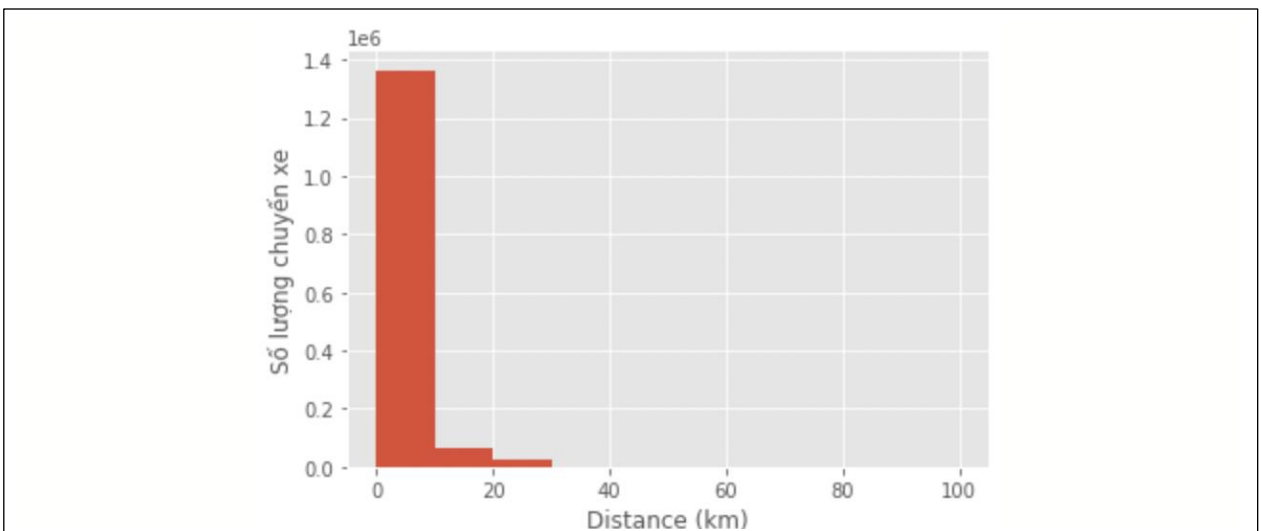
Output[67]:



Quan sát bản đồ, ta thấy các địa điểm trả khách đã phân bố đều hơn, rộng khắp hơn các địa điểm đón khách, lan ra xung quanh các khu vực của NYC. Nhưng ta cũng nhận thấy rằng, Manhattan vẫn là nơi có nhiều chuyến xe tập trung đến trả khách nhất.

Ta sẽ xem phân bố của cột **distance**

Output[68]:

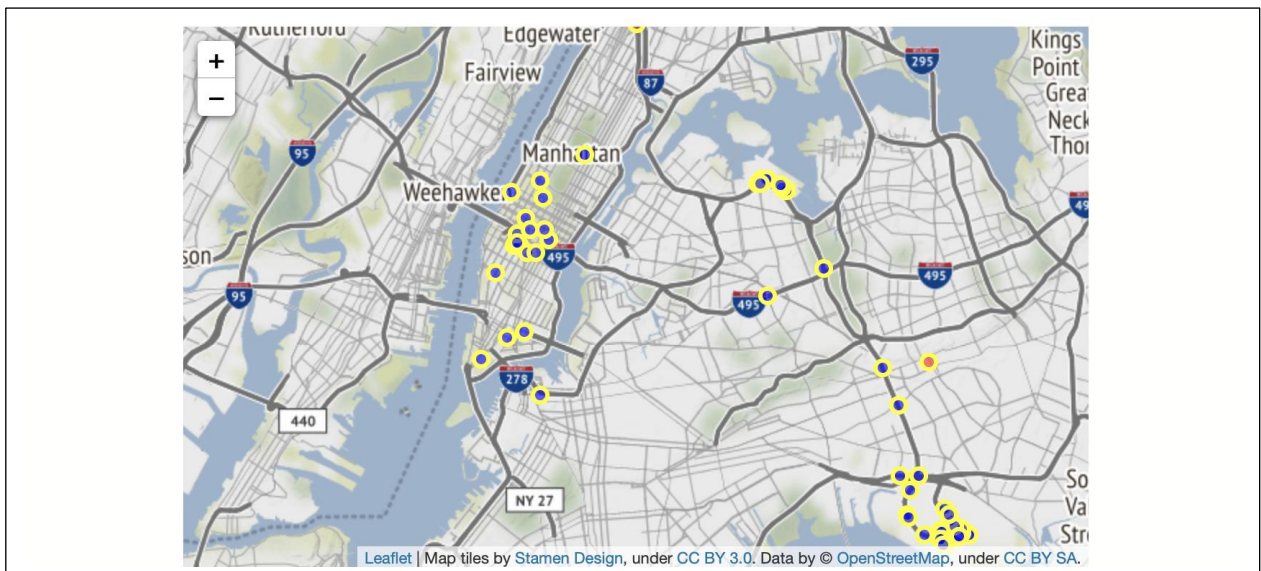


Dựa vào biểu đồ trên, ta có thể thấy phần lớn các chuyến xe trong tập dữ liệu đều có khoảng cách di chuyển dưới 20 km.

Như vậy, ta sẽ tạm định nghĩa chuyến dài là những chuyến có khoảng cách lớn hơn 50km. Ta cũng có thể nhận ra điều này khi xem phân bố của cột distance khi mà phần lớn các chuyến đi đều có khoảng cách di chuyển dưới 50km và trong thực tế để đi một vòng NYC ta cũng sẽ cần đi 1 khoảng là 48.28km.

Ta sẽ xem vị trí các điểm đón của các chuyến đi dài vì có thể các chuyến này sẽ xuất phát từ các sân bay đi vào trung tâm quận Manhattan. Tại đây ta có sử dụng thư viện folium để hỗ trợ vẽ bản đồ phía dưới, cho phép chúng ta thấy rõ được vị trí của các địa điểm quan trọng cần chú ý.

Output[69]:



Quan sát bản đồ, ta thấy điểm đón khách của các chuyến đi dài này tập trung ở 3 khu vực chính là khu Manhattan, sân bay John F. Kennedy và sân bay LaGuardia. Còn các điểm trả khách đa số nằm ngoài phạm vi thành phố New York. Manhattan cũng chiếm đa số điểm đón các chuyến dài

Như vậy, cũng có những chuyến xe xuất phát từ 2 sân bay này và có cự ly di chuyển dài ( $>50$  km), điều này cũng sẽ dẫn đến thời gian di chuyển của chuyến đi sẽ dài hơn, nhưng liệu rằng ta có nên thêm một thuộc tính mới đánh dấu những chuyến xe có điểm đón hoặc điểm trả nằm gần 2 sân bay kể trên để đưa vào tập dữ liệu huấn luyện mô hình hay không?

Có thể tham khảo thêm tài liệu về 2 thư viện đã được sử dụng ở phía trên:

- Thư viện `mpl_toolkits.basemap`: <https://matplotlib.org/basemap/>
- Thư viện `folium`: <https://python-visualization.github.io/folium/>

#### **4. HUẤN LUYỆN MÔ HÌNH DỰ ĐOÁN TỔNG THỜI GIAN DI CHUYỂN CỦA CHUYẾN DI (TRIP\_DURATION).**

Ta sẽ thử chạy các mô hình giống nhau về bản chất, ở đây chúng ta sẽ chọn mô hình XGBoost, để huấn luyện so sánh kết quả giữa các mô hình với data khác nhau để thấy được sự thay đổi của kết quả nếu một hay tất cả thuộc tính vừa được feature engineer tìm ra.

Cuộc thi New York City Taxi Trip Duration đặt ra bài toán dự đoán tổng thời gian di chuyển của một chuyến taxi ở thành phố New York (Mỹ) dựa vào các thông tin như tọa độ điểm đón và trả khách, thời gian đón khách, số lượng hành khách. Với output là tổng thời gian dự đoán của chuyến đó tính bằng giây. Thì chúng ta thử dự đoán và đánh giá kết quả.

Qua việc tìm hiểu và phân tích tập dữ liệu thì chúng ta đã có thêm hiểu biết hơn về dữ liệu. Với những giả thuyết được đưa ra thì chúng ta có những feature tiêu biểu như sau:

- Weekday: Thứ mấy trong tuần, nằm trong đoạn [0, 6].
- Hour: Giờ trong ngày, thuộc khoảng [0,23].
- pickup\_longitude: Vị trí điểm đón.
- pickup\_latitude: Vị trí điểm đón.
- dropoff\_longitude: Vị trí điểm trả.
- dropoff\_latitude: Vị trí điểm trả.
- distance: Khoảng cách giữa điểm đón và trả.
- speed: Tốc độ của chuyến đi.
- Low\_speed\_time: Khung giờ có tốc độ trung bình thấp.

##### **4.1. Tiền xử lý.**

Trước tiên, để làm cho công việc được liền mạch cũng như tránh sai sót, ở đây chúng ta sẽ có một hàm gom lại tất cả các việc đã làm ở phần trước.

Ở phía trên để thuận tiện cho việc đọc các tập Train và Test sau này, ta có sử dụng thư viện pickle để làm việc với file nhị phân. Pickle được sử dụng để thực hiện chuyển đổi các cấu trúc đối tượng Python sang một dạng byte để có thể được lưu trữ trên ổ đĩa hoặc được gửi cho người khác qua mạng. Sau đó, luồng ký tự này sau đó có thể được truy xuất và chuyển đổi trở lại sang dạng đối tượng ban đầu trong Python. Xem thêm tài liệu về thư viện Pickle [tại đây](#).

Để thuận tiện thì ta chỉ cần đọc dữ liệu thông qua file pickle đã được xử lý ở phía trên.

Để so sánh chúng ta sẽ chọn ra 4 tập dữ liệu tiêu biểu khác nhau và sử dụng chung 1 model:

- X1: Dữ liệu gốc bao gồm các cột: weekday, hour, pickup\_longitude, pickup\_latitude, dropoff\_longitude, dropoff\_latitude, passenger\_count, vendor\_id, store\_and\_fwd\_flag.
- X2: Là dữ liệu X1 nhưng có thêm thuộc tính distance.
- X3: Là dữ liệu X1, nhưng có thêm thuộc tính Low\_speed\_time.
- X4: Là dữ liệu X1, nhưng có thêm thuộc tính distance và Low\_speed\_time.

Đối với model XGBoost thì ta cũng dùng XGBoost không tinh chỉnh:

Output[70]: Tập dữ liệu X1

```
[15:14:38]
```

Output[71]: Tập dữ liệu X2

```
[15:16:36]
```

Output[72]: Tập dữ liệu X3

```
[15:18:39]
```







Output[73]: Tập dữ liệu X4

[15:20:42]

## 4.2. Kết quả.

Toàn bộ kết quả dự đoán của 4 mô hình sẽ được upload trên Kaggle để tính private score và public score của cuộc thi. Kết quả như sau:

- Model 1: Public score: 0.65572, Private score: 0.65763
- Model 2: Public score: 0.49539, Private score: 0.49437
- Model 3: Public score: 0.63515, Private score: 0.6374
- Model 4: Public score: 0.49284, Private score: 0.49216

Submission and Description		Private Score ⓘ	Public Score ⓘ	Selected
	<b>XGB_1.csv</b> Complete (after deadline) · 1m ago	<b>0.65763</b>	<b>0.65572</b>	<input type="checkbox"/>
	<b>XGB_2.csv</b> Complete (after deadline) · 1m ago	<b>0.49437</b>	<b>0.49539</b>	<input type="checkbox"/>
	<b>XGB_3.csv</b> Complete (after deadline) · 2m ago	<b>0.6374</b>	<b>0.63515</b>	<input type="checkbox"/>
	<b>XGB_4.csv</b> Complete (after deadline) · 2m ago	<b>0.49216</b>	<b>0.49284</b>	<input type="checkbox"/>

Nhận xét:

- Khi huấn luyện mô hình với tập dữ liệu gốc, kết quả cho ra sai số là cao nhất với public score: 0.65572 và private score: 0.65763
- Thuộc tính distance có sự tương quan với trip\_duration tốt hơn thuộc tính Low\_speed\_time khi train model cho ra kết quả tốt hơn.
- Ta thấy model 4 khi thêm thuộc tính distance và Low\_speed\_time có độ chính xác cao hơn các model còn lại.

Vì vậy khi huấn luyện mô hình ta cần có quá trình tìm hiểu và trích xuất ra các thuộc tính mới (feature engineering) nhằm cải thiện độ chính xác cho model.



## TỔNG KẾT

Dựa vào giai đoạn phân tích dữ liệu ta có thể trích xuất được các thông tin hữu ích như: các khung giờ cao điểm, những vị trí có nhiều lượt đón/trả khách hay các khung giờ có tốc độ di chuyển cao/thấp và từ đó chúng ta có thể đưa ra các quyết định chính xác hơn từ dữ liệu hay bổ sung thêm (hoặc loại bỏ đi) những biến có ích hoặc gây ra bất lợi trong quá trình đào tạo mô hình.

Ngoài ra, chúng ta còn biết thêm một thông tin là lượng đặt xe taxi nói riêng hay nói rộng hơn là mật độ các phương tiện tham gia giao thông sẽ giảm đi vào những ngày có thời tiết khắc nghiệt (bão, mưa tuyết) so với những ngày thường.

Đối với việc thêm thuộc tính mới là cột đánh dấu những chuyến xe có điểm đón hoặc điểm trả nằm gần 2 sân bay vào tập dữ liệu đào tạo mô hình đã nêu ở phần trên, sau khi tiến hành thử nghiệm nhưng kết quả cho ra không có mấy cải thiện và đi kèm với thời gian thực hiện khá lâu, nên nhóm đã quyết định bỏ đi thuộc tính này và chỉ giữ lại những thuộc tính tiêu biểu phía trên để đào tạo mô hình dự đoán.

Ta nhận thấy khi xây dựng một mô hình nào dự đoán nào đó từ dữ liệu, ta không thể chỉ dùng các biến đã có sẵn trong tập dữ liệu để đào tạo mô hình, mà cần phải có một giai đoạn vô cùng quan trọng đó là Feature Engineering. Feature Engineering là một giai đoạn không thể thiếu trong quá trình phát triển bất kỳ một hệ thống thông minh nào. Đây là quá trình biến đổi dữ liệu thành các đặc trưng (các cột/biến) đóng vai trò là đầu vào cho các mô hình học máy, các đặc trưng được xử lý tốt sẽ nâng cao hiệu suất của mô hình.

Sau khi thực hiện xong bài báo cáo, nhóm đã có cơ hội được tìm hiểu và làm việc thêm với những thư viện, kỹ thuật lập trình Python mới, qua đó góp phần trau dồi thêm kỹ năng cho các thành viên thực hiện đề tài báo cáo.