TRƯỜNG ĐẠI HỌC SÀI GÒN

Khoa Công nghệ thông tin

Bài thu hoạch

nhóm Môn học:

Phân tích dữ liệu **Lớp:**

DCT121C3

Giảng viên: PGS. TS. Nguyễn Tuấn Đăng

Họ và Tên các thành viên:

Trần Gia Phú **MSSV**: 3121411166

Năm học: 2023 – 2024 / HK2

MỤC LỤC

Danh Mục Hình Ảnh

Matplotlib

Seaborn

Bokeh

Seaborn

1. Giới thiệu

Mục tiêu:

Có thể hiểu và sử dụng thành thạo các thư viện visualization để trực quan hóa dữ liệu một cách hiệu quả. Có khả năng trình bày và phân tích dataset thu thập được, tìm ra các thông tin có ý nghĩa từ dataset đó. Biết cách đưa ra những nhận định và kết luận từ các dữ liệu được trình bày.

Phạm vi:

Trong bài kì này, chúng ta sẽ tập trung vào sử dụng các gói thư viện như Matplotlib, Seaborn và Bokeh để phân tích và trực quan hóa dữ liệu thông qua các biểu đồ. Đồng thời, chúng ta cũng sẽ sử dụng Pandas và Spark để xử lý và quản lý dữ liệu một cách hiệu quả.

Yêu cầu về cài đặt:

Để thực hiện các dạng biểu đồ, chúng ta cần cài đặt các thư viện sau: Matplotlib, pandas, Seaborn, Bokeh.

Matplotlib:

Cài đặt matplotlib bằng pip:

pip install matplotlib

Đối với Python 3:

pip3 install matplotlib

Và cũng có thể cài đặt từ CMD:

conda install matplotlib

Seaborn:

Cài đặt Seaborn bằng pip:

pip install seaborn

Đối với Python 3:

pip3 install seaborn

Và cũng có thể cài đặt từ CMD:

conda install seaborn

Bokeh:

Cài đặt bằng pip:

pip install bokeh

Cài đặt từ CMD:

conda install bokeh

2. Sơ lược về Dataset:

DataSet: New York City Airbnb Open Data

Nguồn: ***https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open- data***

Dataset " Nyc_Airbnb " là một tập dữ liệu về việc cho thuê nhà trên nền tảng Airbnb tại New York City vào năm 2019 . Tập dữ liệu này cung cấp thông tin về các loại phòng,các host cũng như là vị trí của các căn hộ được đăng ký trên Airbnb tại thành phố New York.

Dưới đây là các thông tin chi tiết của dataset:

id: ID duy nhất của mỗi danh sách phòng.

name: Tên của phòng.

host_id: ID của người chủ sở hữu phòng.

host_name: Tên của người chủ sở hữu phòng.

neighbourhood_group: Nhóm khu vực của phòng.

neighbourhood: Khu vực cụ thể của phòng.

latitude: Vĩ độ của vị trí phòng.

longitude: Kinh độ của vị trí phòng.

room_type: Loại phòng.

price: Giá thuê của phòng.

minimum_nights: Số đêm tối thiểu mà một khách hàng phải đặt ít nhất khi thuê phòng.

number_of_reviews: Số lượng đánh giá của phòng.

last_review: Ngày đánh giá gần nhất của phòng.

reviews_per_month: Số lượng đánh giá trung bình mỗi tháng của phòng.

calculated_host_listings_count: Số lượng phòng được tính toán bởi mỗi chủ sở hữu.

availability_365: Số ngày trong năm mà danh sách có sẵn để thuê

3. Các bước chuẩn bị

Import:

Import các thư viện và modules trong Python để sử dụng trong việc trực quan hóa dữ liệu.

[2]:
```
import matplotlib as mp
from matplotlib.ticker import FuncFormatter
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from bokeh.models import CustomJS, ColumnDataSource, Slider
from bokeh.layouts import row,column
from pyspark.sql import SparkSession
from bokeh.models import ColumnDataSource, CategoricalColorMapper, HoverTool
from matplotlib.ticker import FuncFormatter
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql import functions as F
from bokeh.plotting import figure, output_notebook, show
from bokeh.models import ColumnDataSource,CustomJS,Slider
from pyspark.sql.functions import year, month, avg
from bokeh.models.layouts import TabPanel, Tabs
#tắt những thông báo warning
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
```

Tạo SparkSession, đọc file csv:

```
#tạo SparkSession
spark = SparkSession.builder.appName("air_bnb_nyc").getOrCreate()
df = spark.read.csv("nyc_air_bnb.csv",header=True)
df.count()
```

```
49079
```

Lúc này số lượng là 49079.

Làm sạch dữ liệu :

CLEANING DATA

```
# lọc những giá trị NaN trong dataset
df = df.dropna(subset=['name', 'host_id', 'host_name',
                       'neighbourhood_group', 'neighbourhood', 'latitude', 'longitude',
                       'price','calculated_host_listings_count','minimum_nights','last_review'])
#lọc những giá trị price > 0
df = df.filter(df['price'] >0)
# lọc những giá trị đêm tối thiểu lớn hơn 0
df = df.filter(df['minimum_nights'] >0)
# Vì 3 phòng này chiếm gần như cả tập dataset nên mình sẽ lọc ra 3 phòng này để phân tích
df =df.filter((col('room_type') == "Shared room") | (col('room_type') == "Private room") | (col('room_type') == "Entire home/apt"))
```

Làm sạch dữ liệu bằng dropna,filter.

Chuyển kiểu dữ liệu:

Chuyển kiểu dữ liệu

```
df = df.withColumn("id", df["id"].cast("int"))\
                .withColumn("name", df["name"].cast("string"))\
                .withColumn("host_id", df["host_id"].cast("int"))\
                .withColumn("host_name", df["host_name"].cast("string"))\
                .withColumn("neighbourhood_group", df["neighbourhood_group"].cast("string"))\
                .withColumn("neighbourhood", df["neighbourhood"].cast("string"))\
                .withColumn("latitude", df["latitude"].cast("float"))\
                .withColumn("longitude", df["longitude"].cast("float"))\
                .withColumn("room_type", df["room_type"].cast("string"))\
                .withColumn("price", df["price"].cast("int"))\
                .withColumn("minimum_nights", df["minimum_nights"].cast("int"))\
                .withColumn("number_of_reviews", df["number_of_reviews"].cast("int"))\
                .withColumn("last_review", to_date(df["last_review"], "yyyy-MM-dd"))\
                .withColumn("reviews_per_month", df["reviews_per_month"].cast("float"))\
                .withColumn("calculated_host_listings_count ", df["calculated_host_listings_count"].cast("int"))\
                .withColumn("availability_365", df["availability_365"].cast("int"))
```

Sử dụng withColumn để chuyển đổi kiểu dữ liệu của thuộc tính.

4. Matplotlib

Để thực hiện các suy luận thống kê cần thiết, cần phải trực quan hóa dữ

liệu và Matplotlib là một trong những giải pháp như vậy cho người dùng Python. Nó là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với Python và NumPy. Module được sử dụng nhiều nhất của Matplotib là Pyplot.

a. Accessories for charts

Trong module matplotlib, chúng ta có thể thêm tiêu đề và nhãn trục cho một biểu đồ. Chúng ta có thể thêm tiêu đề bằng cách sử dụng plt.title() và nhãn bằng cách sử dụng plt.xlabel() và plt.ylabel(). Nhiều biểu đồ có nghĩa là nhiều đối tượng, như đường, cột, và phân tán.

```
df_private_room = df.filter(df['room_type'] == 'Private room')
df_entire_home = df.filter(df['room_type'] == 'Entire home/apt')
# Tạo cột mới 'year' từ cột 'last_review' để lấy năm của người đánh giá
df_private_room_with_year = df_private_room.withColumn('year', year(to_date(df_private_room['last_review'])))
df_entire_home_with_year = df_entire_home.withColumn('year', year(to_date(df_entire_home['last_review'])))
# Tính giá trị trung bình của 'number_of_reviews' theo năm cho phòng riêng tư (Private room)
average_reviews_private_room_by_year = df_private_room_with_year.groupby('year').avg('number_of_reviews').orderBy('year')

# Tính giá trị trung bình của 'number_of_reviews' theo năm cho toàn bộ căn hộ (Entire Home/apt)
average_reviews_entire_home_by_year = df_entire_home_with_year.groupby('year').avg('number_of_reviews').orderBy('year')

# Chuyển đổi kết quả sang Pandas DataFrame để vẽ biểu đồ
average_reviews_private_room_by_year_pd = average_reviews_private_room_by_year.toPandas()
average_reviews_entire_home_by_year_pd = average_reviews_entire_home_by_year.toPandas()

# Vẽ biểu đồ đường
plt.plot(average_reviews_private_room_by_year_pd['year'], average_reviews_private_room_by_year_pd['avg(number_of_reviews)'], marker='o', linestyle='-', label='Private Room')
plt.plot(average_reviews_entire_home_by_year_pd['year'], average_reviews_entire_home_by_year_pd['avg(number_of_reviews)'], marker='o', linestyle='-', label='Entire Home/apt')

plt.xlabel('Năm')
plt.ylabel('Lượng Đánh Giá Trung Bình')
plt.title('Đánh Giá Trung Bình Theo Các Năm')
plt.legend()
plt.grid(True)
# Hiển thị đồ thị
plt.show()
```

Ở đoạn code trên để sử dụng plot ta sử dụng *plot()* để điều chỉnh thì ta truyền tham số :

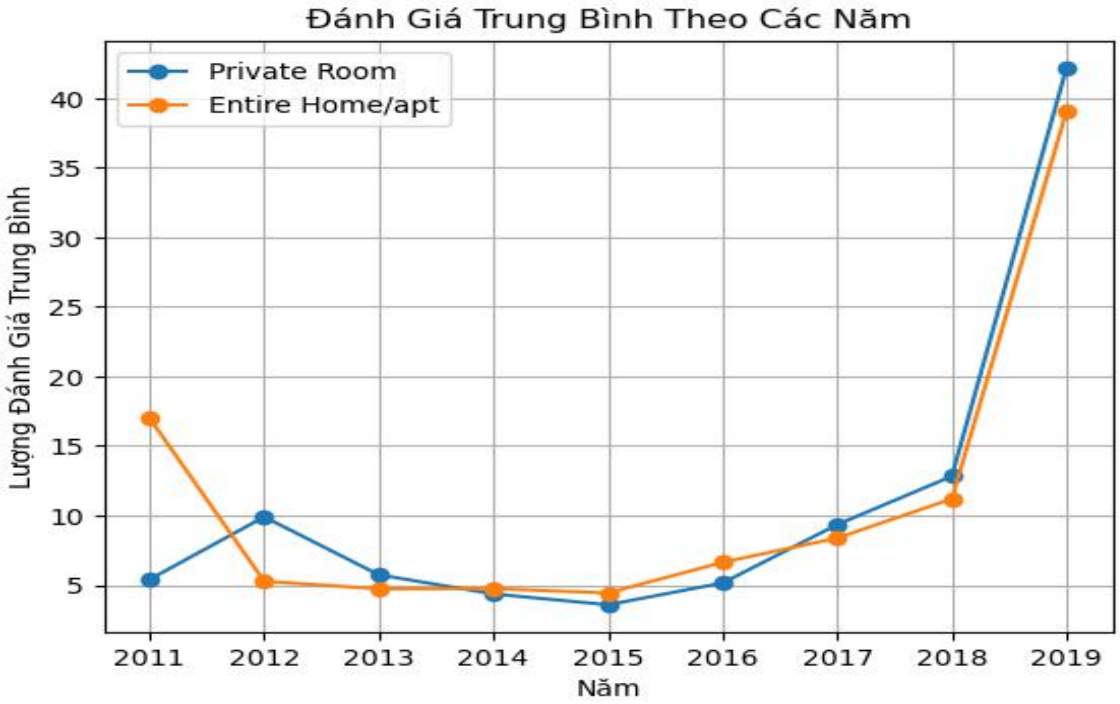
x và y là số năm và số reviews trung bình

marker là để điều chỉnh các giá trị hiện lên

linestyle là đường để kết nối các marker

label là tên loại phòng

Hình minh họa:



Hình 4.0 – Biểu đồ Plot đánh giá trung bình theo các năm của private room và entire home/apt

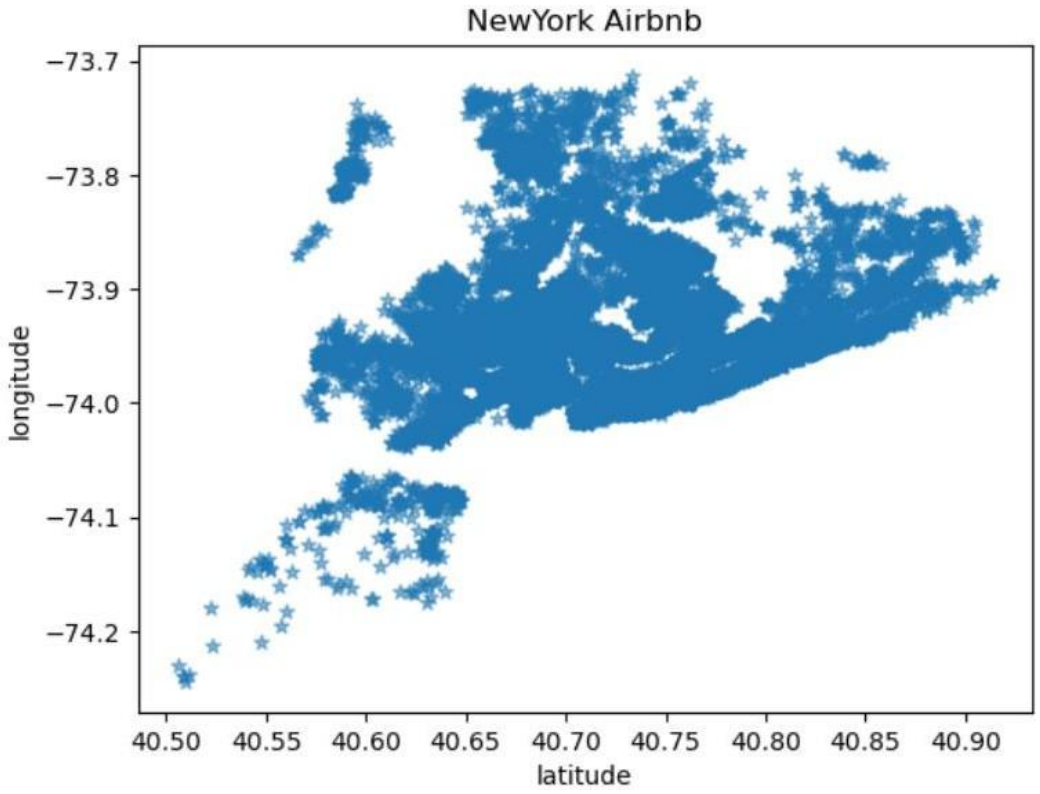
b. Scatter plot

Scatter Plot (Biểu đồ phân tán) vẽ các điểm dữ liệu sử dụng tọa độ Descartes để hiển thị giá trị của các biến dữ liệu số. Thể hiện mối quan hệ giữa hai giá trị số. Ta có thể biểu diễn một biểu đồ phân tán trong Matplotlib bằng cách sử dụng hàm *scatter().*

```
#Scatter
lat_long=plt.scatter(data=Lat_long_type, x='latitude', y='longitude',marker='*',alpha=0.5)
#alpha: là mức độ trong suốt của các marker sẽ có giá trị nằm trong khoảng 0 tới 1
plt.title('NewYork Airbnb')
plt.ylabel("longitude")
plt.xlabel("latitude")
plt.show()
```

Trong đoạn code trên,trong *scatter()* sẽ lấy giá trị của Lat_long_type, với cột x là 'latitude' và cột y là 'longitude' và điều chỉnh các thuộc tính cho biểu đồ: marker, alpha.

Hình minh họa:



Hình 4.1 – Biểu đồ scatter phân bố các Airbnb ở NewYork

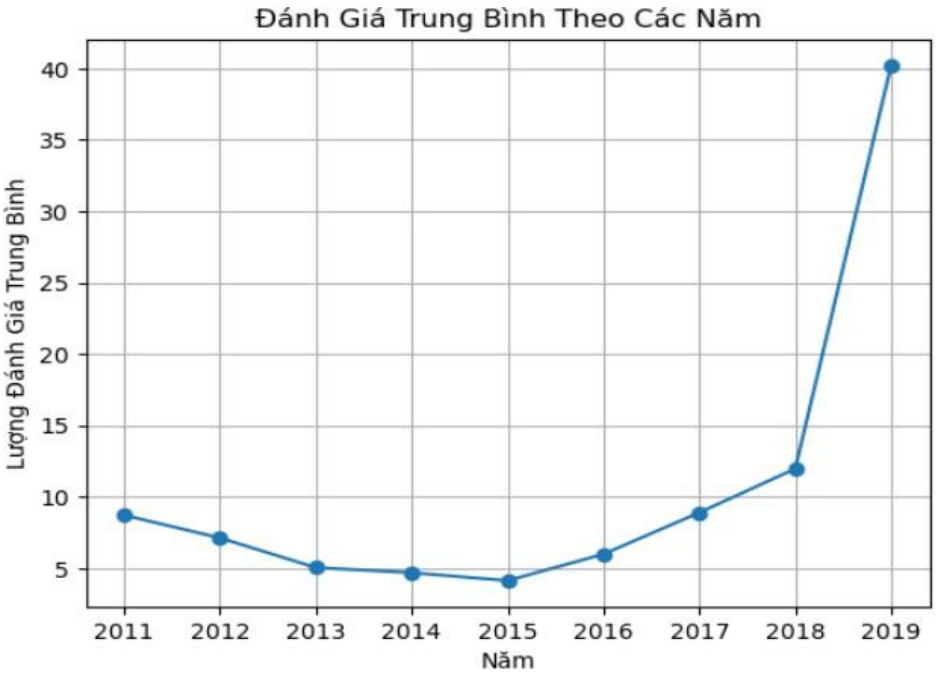
c. Line plot

Line plot (Biểu đồ đường) là biểu đồ biểu thị một đường thẳng giữa 2 biến số. Nó có một chuỗi điểm dữ liệu được nối bởi một đoạn thẳng.

```
## LINE PLOT

# Tạo cột mới 'year' từ cột 'last_review' để lấy năm của người đánh giá
airbnb_with_year = df.withColumn('year', year(to_date(df['last_review'])))

# Tính giá trị trung bình của 'number_of_reviews' theo năm
average_reviews_by_year = airbnb_with_year.groupby('year').avg('number_of_reviews').orderBy('year')

# Chuyển đổi kết quả sang Pandas DataFrame để vẽ biểu đồ
average_reviews_by_year_pd = average_reviews_by_year.toPandas()

# Vẽ biểu đồ đường
plt.plot(average_reviews_by_year_pd['year'], average_reviews_by_year_pd['avg(number_of_reviews)'], marker='o', linestyle='-')
plt.xlabel('Năm')
plt.ylabel('Lượng Đánh Giá Trung Bình')
plt.title('Đánh Giá Trung Bình Theo Các Năm')
plt.grid(True)

# Hiển thị đồ thị
plt.show()
```

Trong đoạn code trên, ta sử dụng *plot()* với 2 điểm x là năm , điểm y là số đánh giá trung bình và điều chỉnh các thuộc tính của biểu đồ bằng: marker, linestyle.

Hình minh họa :



Hình 4.2 Biểu đồ Line plot về số đánh giá trung bình của các năm

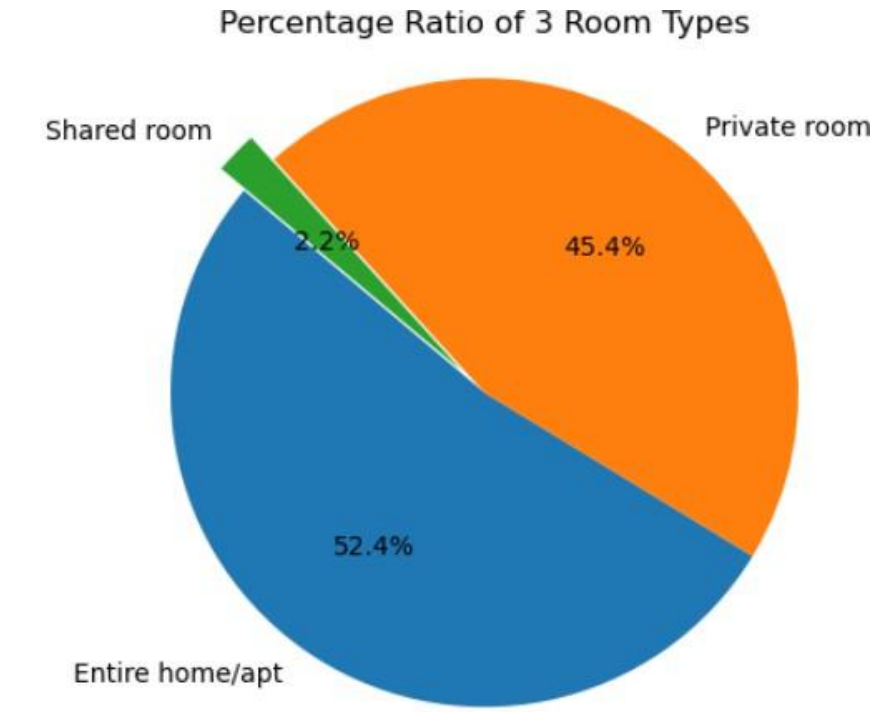
d. Pie plot

Pie plot (Biểu đồ tròn) là một đồ thị tròn được chia thành các phần hình lê. Mỗi phần tử đại diện cho một giá trị và tỉ lệ với giá trị đó. Tổng giá trị của biểu đồ tròn là 100%.

```
# Tính toán số lượng các loại phòng
room_type_counts = df.groupBy("room_type").count()

# Sắp xếp theo số lượng giảm dần
room_type_counts = room_type_counts.orderBy('count', ascending=False)

# Lấy 3 loại phòng
top_room_types = room_type_counts.limit(3).collect()

# Chuẩn bị dữ liệu cho biểu đồ Pie
labels = [row['room_type'] for row in top_room_types]
counts = [row['count'] for row in top_room_types]
explode = (0, 0, 0.1)  # Đặt giá trị tách ra cho 3 phần đầu tiên, các phần còn lại không tách ra

# Vẽ biểu đồ Pie
plt.pie(counts, labels=labels, autopct='%1.1f%%', startangle=140, explode=explode)
plt.axis('equal')  # Ensure the pie chart is circular
plt.title('Percentage Ratio of 3 Room Types')
# plt.legend()
plt.show()
```

Tham số startangle chỉ định góc bắt đầu, là 140 độ và tự động định dạng phần trăm. Tham số explode để kéo ra một phần của bánh theo giá trị nhị phân.

Hình minh họa:



Biểu đồ này cho thấy rằng:

- Phòng Entire home/apt chiếm 52% tổng số
- Phòng Private Room chiếm 45.6% tổng số
- Phòng Share home chiếm 2.4% tổng số

Hình 4.3 – Biểu đồ Bar plot về tỉ lệ phần trăm của 3 loại phòng

e. Bar plot

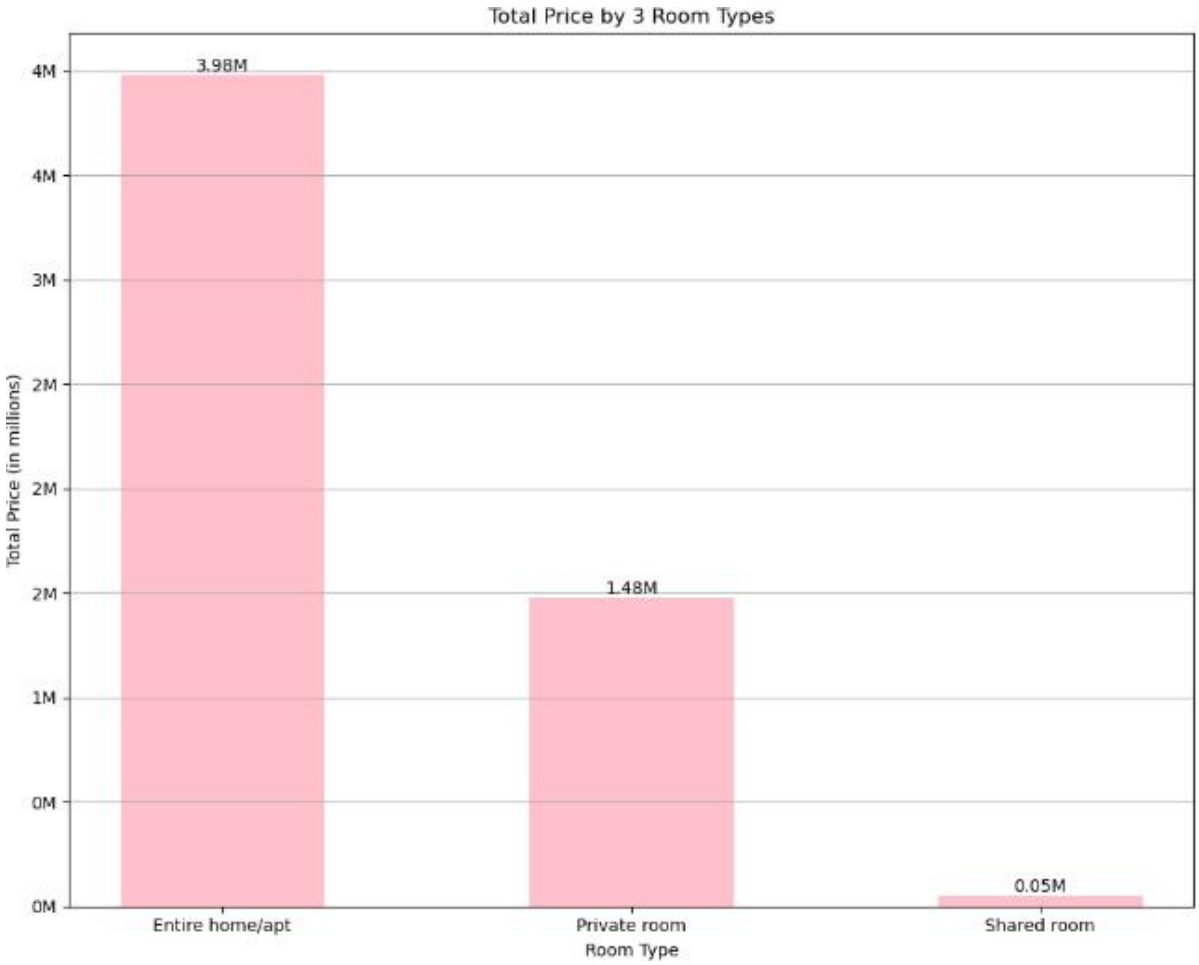
```
total_price_by_room_type = df.groupBy('room_type').sum('price')
total_rate_by_room_type = df.groupBy('room_type').sum('number_of_reviews')
```

Bar plot (Biểu đồ cột) là một công cụ trực quan để so sánh các giá trị của các nhóm khác nhau. Nó có thể được vẽ theo chiều ngang hoặc dọc. Chúng ta có thể tạo ra một biểu đồ cột bằng cách sử dụng hàm *bar().*

```
plt.figure(figsize=(20, 8))

plt.subplot(1, 2, 1)

#chuyển thành dataframe
total_price_by_room_type_pandas = total_price_by_room_type.toPandas()

# lấy ra 3 type phòng có tổng lớn nhất
top_room_types = total_price_by_room_type_pandas.nlargest(3, 'sum(price)')
bars_price = plt.bar(top_room_types['room_type'], top_room_types['sum(price)'], color='pink', width=0.5)

formatter_price = FuncFormatter(lambda x, _: '{:.0f}M'.format(x / 1000000))
plt.gca().yaxis.set_major_formatter(formatter_price)

plt.xlabel('Room Type')
plt.ylabel('Total Price (in millions)')
plt.title('Total Price by 3 Room Types')
plt.grid(axis='y')

# thêm nội dung bên trên bar
for bar in bars_price:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height, '{:.2f}M'.format(height / 1000000),
             ha='center', va='bottom')

# Second subplot:(số hàng,số cột,chỉ số trục con)
plt.subplot(1, 2, 2)
#chuyển thành dataframe
total_rate_by_room_type_pandas = total_rate_by_room_type.toPandas()

# lấy ra 3 type phòng có tổng lớn nhất
top_room_types = total_rate_by_room_type_pandas.nlargest(3, 'sum(number_of_reviews)')
bars_rate = plt.bar(top_room_types['room_type'], top_room_types['sum(number_of_reviews)'], color='pink', width=0.5)

formatter_rate = FuncFormatter(lambda x, _: '{:.0f}'.format(x))
plt.gca().yaxis.set_major_formatter(formatter_rate)

plt.xlabel('Room Type')
plt.ylabel('Total Rate')
plt.title('Total Rate by 3 Room Types')
plt.grid(axis='y')

# thêm nội dung bên trên bar
for bar in bars_rate:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height, '{:.0f}'.format(height),
             ha='center', va='bottom')

plt.tight_layout()  # đảm bảo 2 biểu đồ k đè nhau
plt.show()
```

Trong chương trình trên, hàm *bar()* nhận các giá trị trục x, giá trị trục y và màu sắc. Chúng ta cũng có thể chỉ định màu sắc của các cột trong biểu đồ cột bằng cách sử dụng tham số màu.

Hình minh họa:

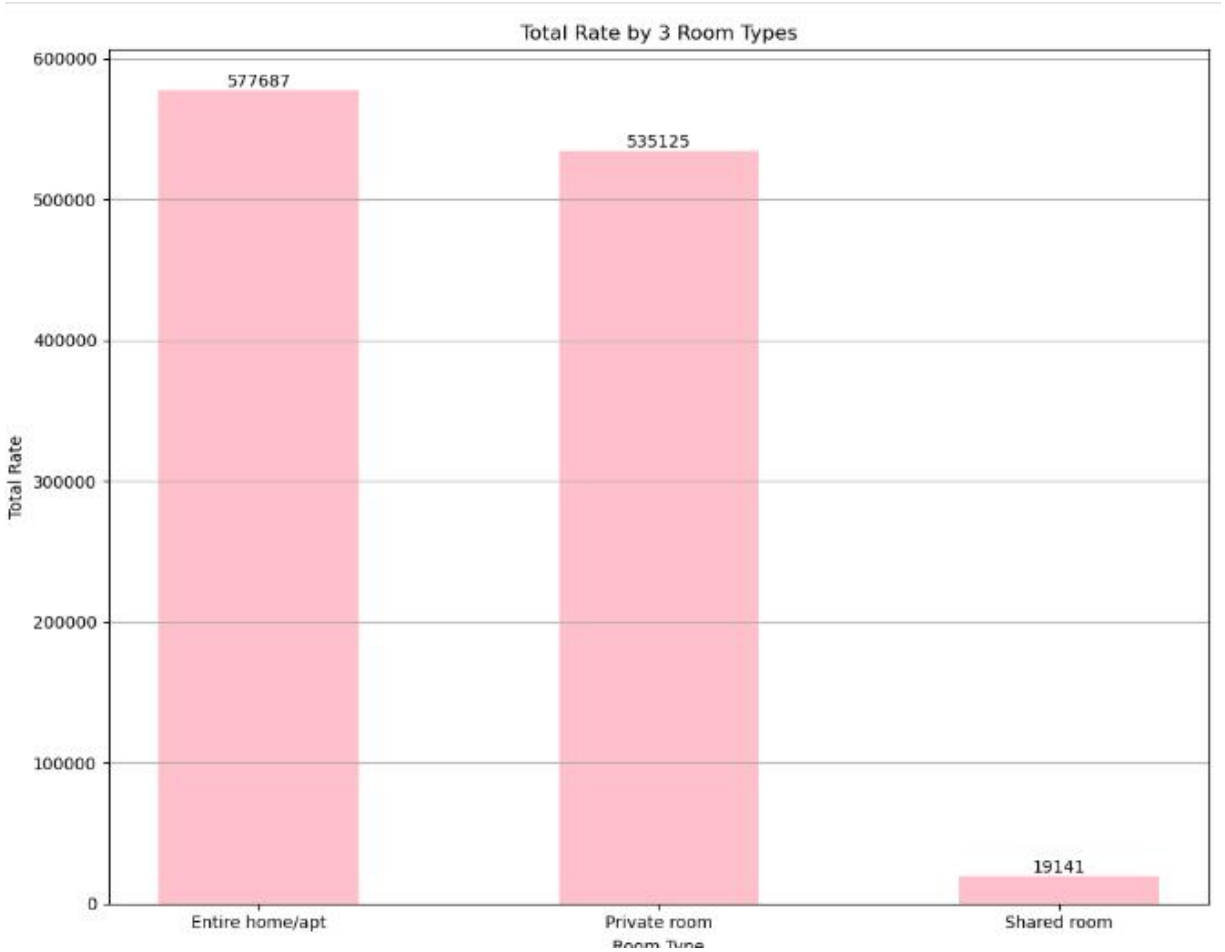


Hình 4.4 – Biểu đồ Bar plot về tổng doanh thu của từng loại phòng

Phòng Entire home/apt có số tiền vượt hơn so với 2 phòng còn lại với tổng doanh thu là 5.37M

Phòng Private Room chiếm tổng doanh thu là 1.99M

Phòng Share home chiếm tổng doanh thu là 0.08M



Hình 4.5 - Biểu đồ Bar plot về tổng lượt đánh giá của từng loại phòng

Tuy là EntireHome/apt có tổng doanh thu chênh lệch với PrivateRoom nhưng về đánh giá thì PrivateRoom vẫn ngang với Entire home/apt.

Nên là về chất lượng đối với khách hàng thì Private room vẫn đạt kết quả tốt như với Entire Home/apt mặc dù ít hơn về mặt số lượng thuê**.**

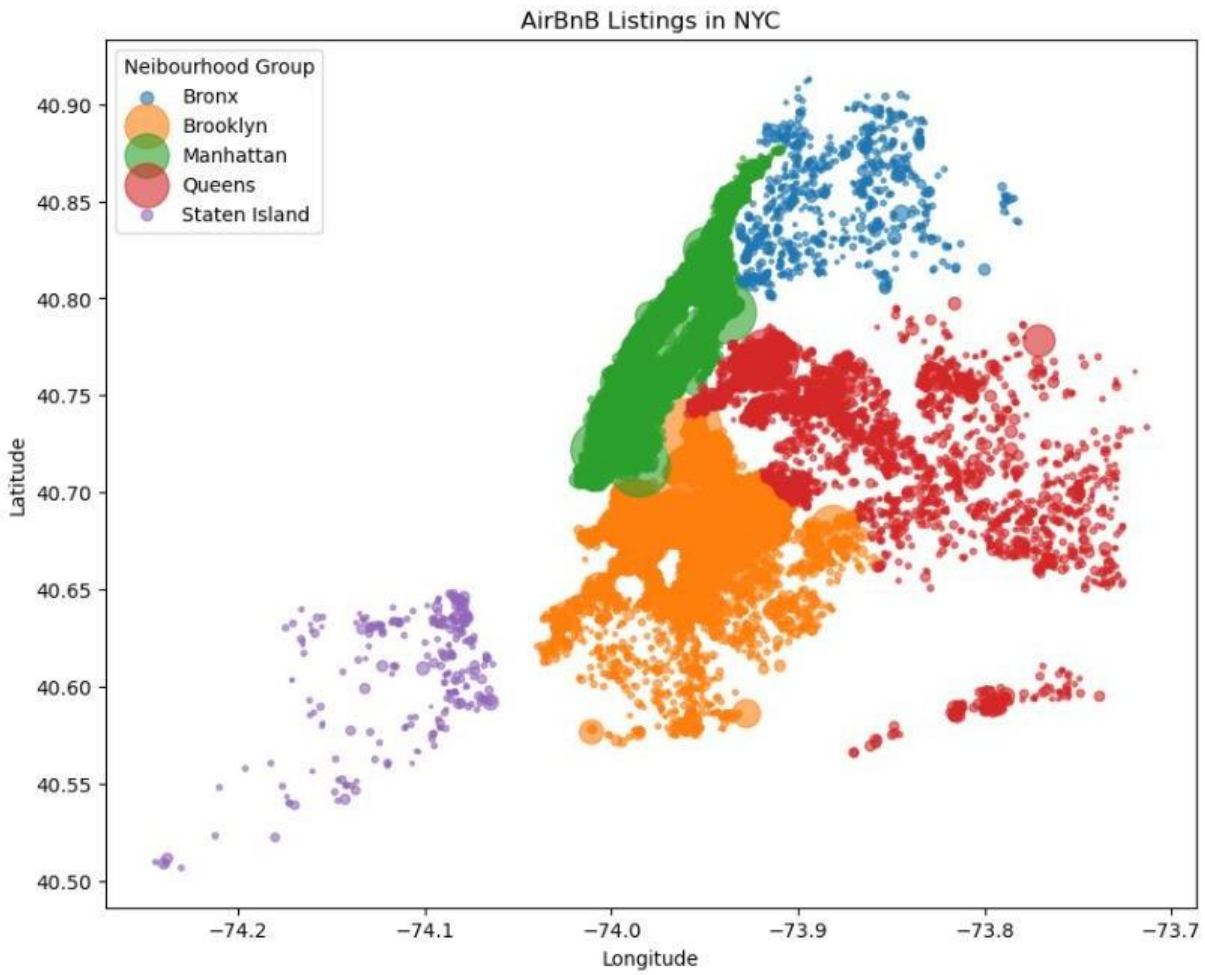
Bubble plot

Bubble plot là một loại biểu đồ phân tán. Nó không chỉ vẽ các điểm dữ liệu bằng các tọa độ Descartes mà còn tạo ra các bong bóng trên các điểm dữ liệu. Bong bóng thể hiện chiều thứ ba của một biểu đồ. Nó hiển thị ba giá trị số: hai giá trị trên các trục x và y và giá trị thứ ba là kích thước của các điểm dữ liệu (hoặc bong bóng).

```
import matplotlib.pyplot as plt

# Giới hạn số lượng hàng mà bạn muốn lấy từ DataFrame
sample_df = df.limit(100000).toPandas()

# Tạo biểu đồ bubble plot
plt.figure(figsize=(10, 8))

# Vẽ các điểm trên biểu đồ với kích thước và màu sắc phụ thuộc vào giá trị từ DataFrame
for group_name, group_df in sample_df.groupby('neighbourhood_group'):
    plt.scatter(group_df['longitude'], group_df['latitude'], s=group_df['price'].astype(float)*0.1, label=group_name, alpha=0.6)

# Thêm tiêu đề và nhãn
plt.title('AirBnB Listings in NYC')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend(title='Neibourhood Group')

# Hiển thị biểu đồ
plt.show()
```

Trong đoạn mã trên, một biểu đồ bubble được tạo ra bằng cách sử dụng hàm scatter. Trong biểu đồ bubble trước này, các giá trị longtitude nằm trên trục x, latitude nằm trên trục y, và price ở từng neibourhood_group được hiển thị bằng kích thước của điểm phân tán hoặc bong bóng. Chúng ta cũng gán một màu ngẫu nhiên cho các bong bóng để làm cho nó hấp dẫn và dễ hiểu hơn.

Hình minh họa:



Hình 4.7 – Biểu đồ Bubble plot về danh sách các nhà trọ ở New York City

Số lượng nhà trọ có ở các quận như Brooklyn, Queens và Manhattan chiếm tỉ lệ khá cao kèm theo với nhiều nhà trọ có giá cao.

Biểu đồ cũng có thể hiện tọa độ của các nhà trọ ở các quận, chúng thể hiện gần như một bản đồ.

f. Pandas plotting

Thư viện pandas cung cấp phương thức plot() như một bọc xung quanh thư viện Matplotlib. Phương thức plot() cho phép chúng ta tạo biểu đồ trực tiếp trên DataFrame của pandas. Các tham số của phương thức plot() sau được sử dụng để

tạo các biểu đồ:

- kind: Một tham số string cho loại biểu đồ, như line, bar, barh, hist, box, KDE, pie, area hoặc scatter.
- figsize: Định nghĩa kích thước cho một hình ảnh trong một tuple (width, height).
- title: Định nghĩa tiêu đề cho biểu đồ.
- grid: Tham số boolean cho các đường lưới trên trục.
- legend: Định nghĩa chú thích.
- xticks: Định nghĩa dãy các dấu chấm trên trục x.
- yticks: Định nghĩa dãy các dấu chấm trên trục y.

```
limit_brooklyn = sum_in_Brooklyn.limit(5)
best_neighbour_in_Brooklyn=Brooklyn.toPandas()

best_neighbour_in_Brooklyn.plot(kind='scatter', x='neighbourhood', y='sum(number_of_reviews)', color='red', title='Số reviews của 5 quận cao nhất ở Brooklyn')
# Show figure
plt.show()
```

Ở đoạn code trên, ta sử dụng plot() với các tham số bên trong là :

kind=’scatter’: sẽ tạo ra các giá trị scatter(chấm)

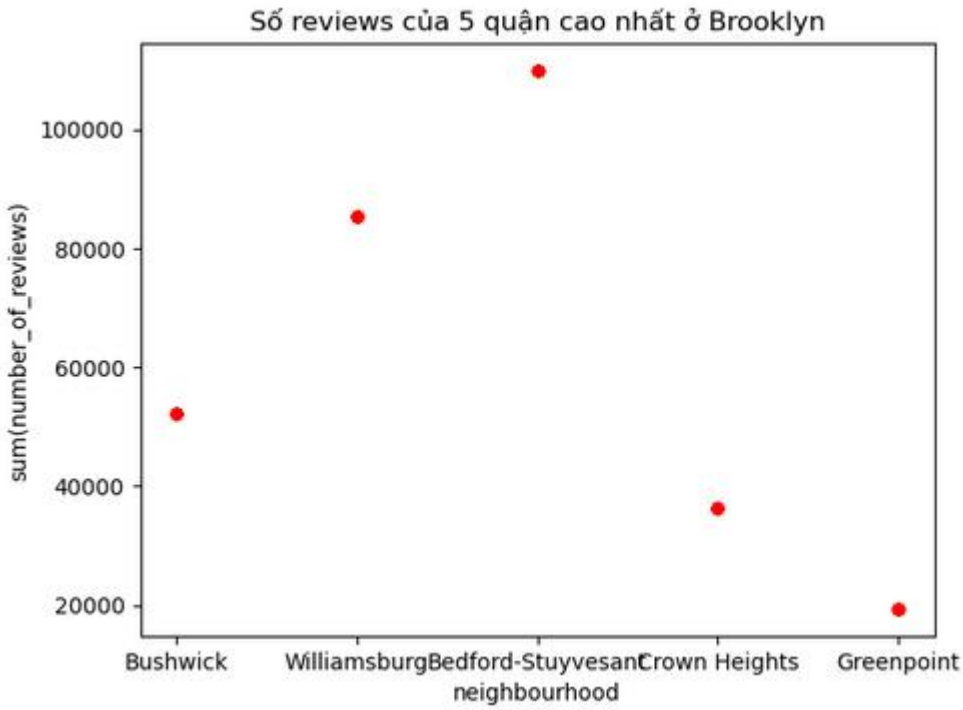
x : hàng x là thuộc tính neighbourhood

y: cột y là thuộc tính tổng của số lượng reviews

color : color cho scatter sẽ là đỏ

title : tên của biê đồ là Số reivews của 5 quận cao nhất ở Brooklyn

Hình minh họa:



Hình 4.8 – Biểu đồ Pandas plot về tổng reviews của 5 quận cao nhất ở Brooklyn

5. Seaborn

Seaborn là một thư viện vẽ biểu đồ dựa trên Matplotlib, được thiết kế để giúp người dùng vẽ các biểu đồ phức tạp một cách dễ dàng hơn. Seaborn cho phép người dùng vẽ các biểu đồ phân tán với các đường hồi quy và các biểu đồ phân phối với các đường cong phù hợp.

```
lat_long=sns.scatterplot(data=Lat_long_type, x='latitude', y='longitude',hue='room_type',marker='*',alpha=0.5)
#alpha: là mức độ trong suốt của các marker sẽ có giá trị nằm trong khoảng 0 tới 1
lat_long.set_title('Sự phân bố của các phòng')
```

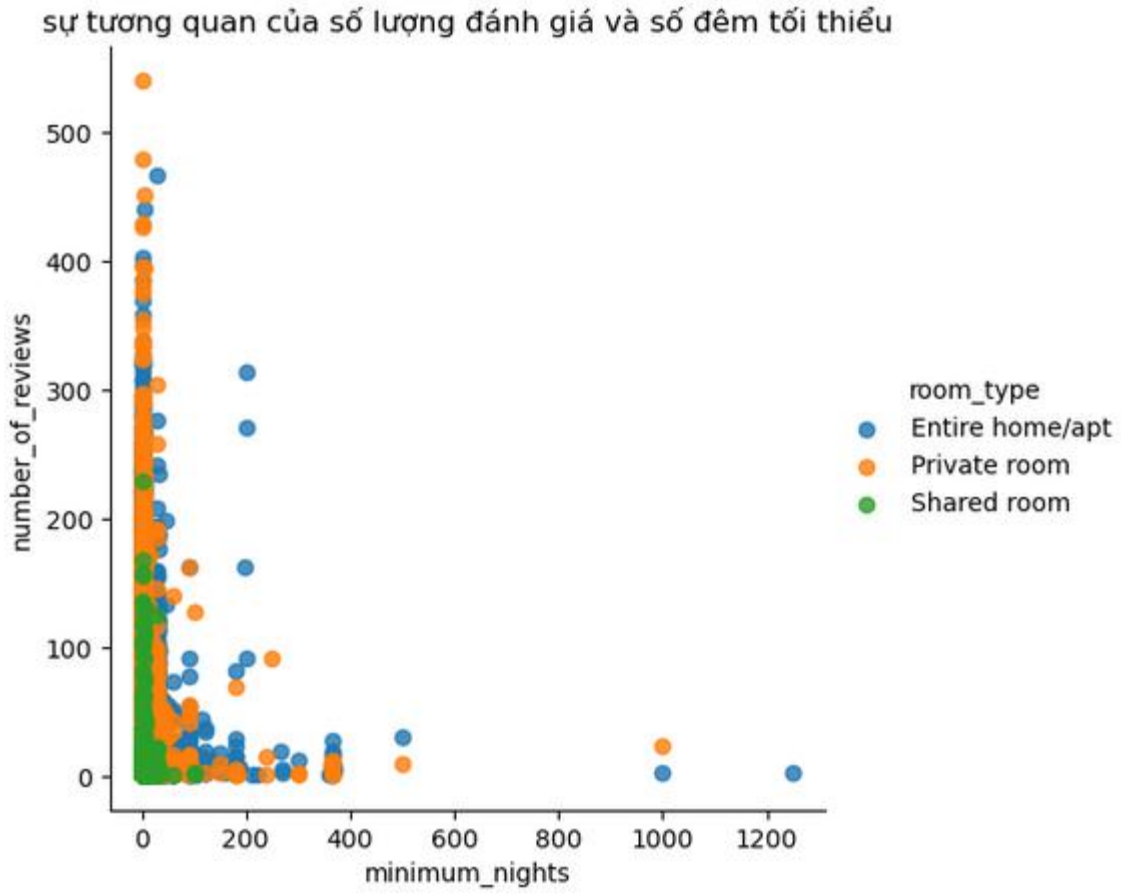
a. lm plots

Hàm lmplot() vẽ biểu đồ phân tán và phù hợp với mô hình hồi quy trên đó. Biểu đồ phân tán là cách tốt nhất để hiểu mối quan hệ giữa hai biến số. Hình ảnh trực quan hóa đầu ra là phân phối chung của hai biến số. Hàm lmplot() nhận hai tên cột - x và y - dưới dạng chuỗi và biến DataFrame.

```
df_pandas = df.toPandas()
df_pandas = df_pandas[df_pandas['room_type'].isin(df_pandas['room_type'].unique()[:3])]
df_filtered = df_pandas[df_pandas['minimum_nights'] > 1]

# Vẽ biểu đồ lmplot với ba giá trị room_type đầu tiên
sns.lmplot(x='minimum_nights', y='number_of_reviews', data=df_filtered, fit_reg=False, hue='room_type')
sns.lmplot
# Hiển thị biểu đồ
plt.title("sự tương quan của số lượng đánh giá và số đêm tối thiểu")
plt.show()
```

lmplot() sẽ vẽ biểu đồ 'minimum_nights' trên trục x, 'number_of_reviews' trên trục y, và màu sắc được phân loại theo 'room_type'.

Hình minh họa:



Biểu đồ trên cho thấy :

- Sự tương quan giữa number_of_reviews và minimum_nights
- Nghĩa là với số đêm tối thiểu càng ít,số lượng đánh giá càng cao
- Có thể lí giải là đa số khách hàng thích lưu trú ngắn hạn
- Và cũng cho thấy được phòng Private Room có được nhiều lượt đánh giá cao nhất

Hình 5.1 – Biểu đồ lm plot về sự tương quan của số lượng đánh giá và số đêm tối thiểu

b. Bar plots

barplot() cung cấp mối quan hệ giữa một biến phân loại và một biến liên

```
# đếm số lượng listings và nhóm theo host_name
sum_by_host = df.groupBy("host_name").agg({"calculated_host_listings_count": "count"})
#Xếp lại từ cao tới thấp của số lượng listings
sum_by_host_sorted = sum_by_host.orderBy(col("count(calculated_host_listings_count)").desc())
#Chọn ra 6 người có số lượng listing cao nhất
sum_by_host_sorted_top6=sum_by_host_sorted.limit(6)
```

```
df_pandas = sum_by_host_sorted_top6.toPandas()
#Sử dụng barplot với giá trị hàng x là host_name và cột y là số lượng listing
ax = sns.barplot(x='host_name', y='count(calculated_host_listings_count)', data=df_pandas)

# Thêm giá trị vào label
for p in ax.patches: # lặp qua từng thanh cột
    height = p.get_height() # lấy chiều cao của các cột
    ax.text(p.get_x() + p.get_width() / 2., height + 3, f"{height:,}", ha='center') # xác định tọa độ x và y để những chú thích ở giữa

# Customize plot
plt.xlabel('Host Name')
plt.ylabel('Total Calculated Host Listings Count')
plt.title('Top 6 Hosts with Highest Calculated Host Listings Count')
plt.tight_layout()
plt.show()
```

tục. Nó sử dụng các thanh hình chữ nhật có độ dài thay đổi.

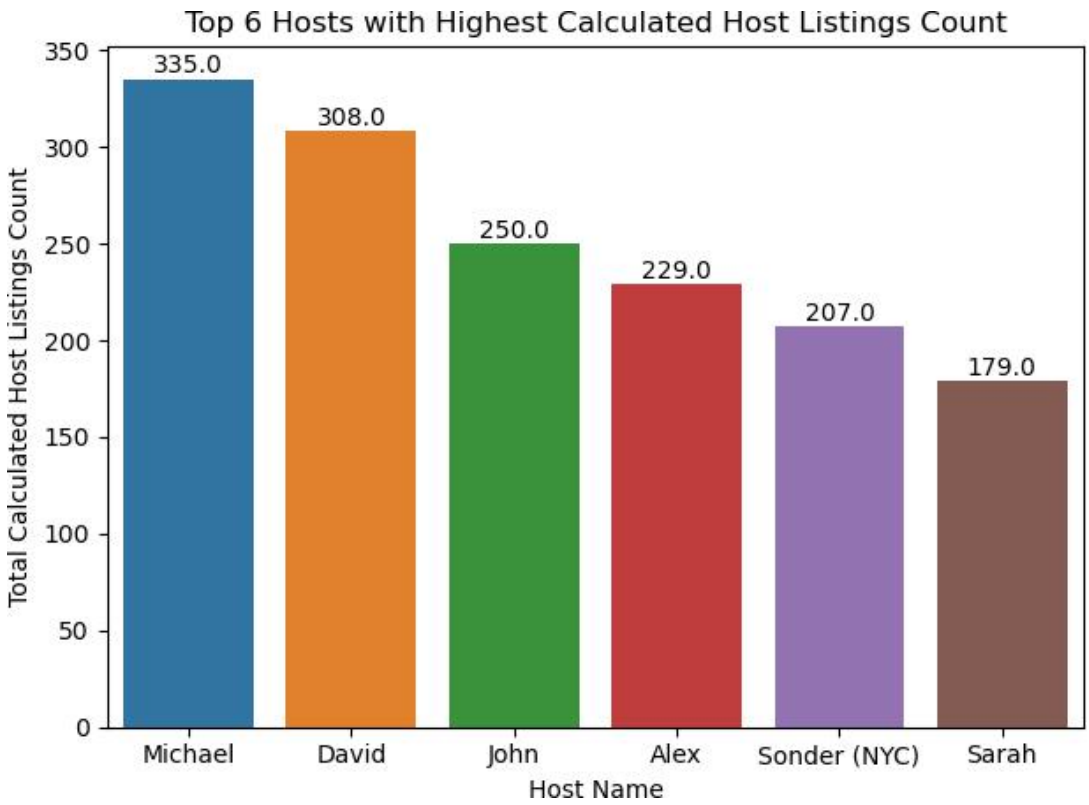
Cột host_name được sử dụng làm trục x

Cột count(calculated_host_listings_count) làm trục y

Dữ liệu được lấy từ DataFrame df_pandas

Sau khi vẽ biểu đồ, một vòng lặp được sử dụng để thêm giá trị của mỗi cột vào các nhãn của cột tương ứng trên biểu đồ.

Hình minh họa:



Hình 5.2 – Biểu đồ Bar plots về Top 6 Host có nhiều phòng nhất ở New York City

Biểu đồ mô tả số lượng các host có số lượng danh sách nhà ở cao nhất được tính toán.

Sự tập trung danh sách này có thể cho thấy rằng những host này là những người có uy tín và giàu kinh nghiệm trong thị trường cho thuê Airbnb ở NY.

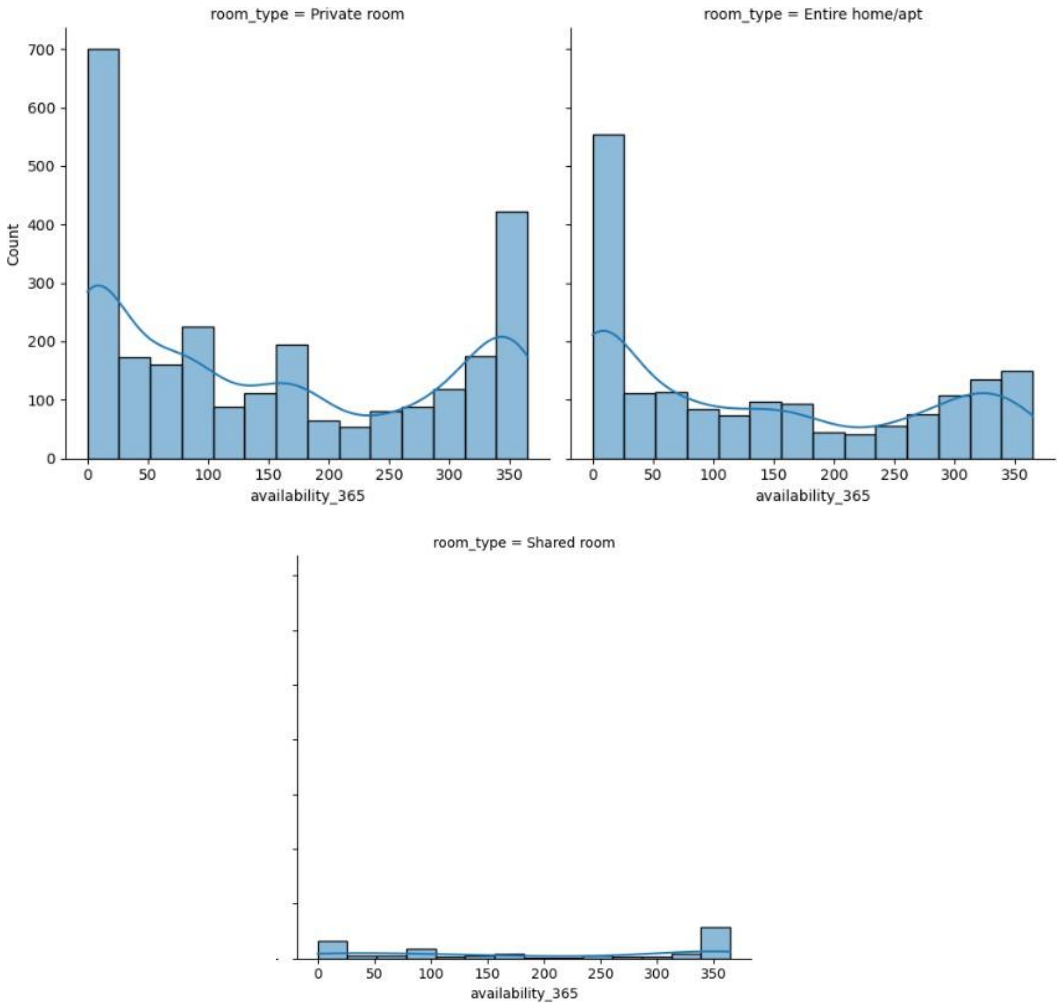
c. Distribution plots

Hàm *distplot()* vẽ một univariate distribution. Nó là sự kết hợp giữa một Histogram với kích thước bin mặc định và một đồ thị KDE.

```
sns.displot(data=Queens_neig_pd,x="availability_365",col="room_type",kde=True)
```

Hiển thị phân phối của cột availability_365 cho từng loại phòng trong cột room_type. Tham số kde=True chỉ định rằng biểu đồ cũng sẽ bao gồm đường KDE để biểu diễn phân phối dạng liên tục.

Hình minh họa:



Hình 5.3 - Biểu đồ Distribution plot về Thống kê số ngày có sẵn và số lượng của các loại phòng tại thành phố Queens

Distribution plot sẽ cho thấy rõ hơn về số lượng các type phòng mà Queens có, dựa trên ngày có sẵn của từng loại phòng. Ở đây cho thấy 2 loại phòng Private và Entire sẽ chiếm đa số về số lượng.

d. Box plots

Box plot (Biểu đồ hộp), còn được gọi là biểu đồ hộp-nhím, là một trong những biểu đồ tốt nhất để hiểu về phân phối của mỗi biến số cùng với các phân vị của nó. Biểu đồ này hiển thị phân phối phân vị trong một hộp,

```
# lấy thành phố có nhiều lượt reviews nhất
max_neighbour_group = sum_by_views.orderBy(F.desc("sum(number_of_reviews)")).first()["sum(number_of_reviews)"]
filtered_df = sum_by_views.filter(sum_by_views["sum(number_of_reviews)"] == max_neighbour_group)
# Hiển thị kết quả
result_df = filtered_df.join(df, "neighbourhood_group", "inner")
# tiếp tục lấy 10 quận cao nhất về số lượng reviews ở Brooklyn
sum_by_reviews_in_Brooklyn = result_df.groupBy("neighbourhood").agg({"number_of_reviews": "sum"})
sum_in_Brooklyn = sum_by_reviews_in_Brooklyn.orderBy(col("sum(number_of_reviews)").desc())
limit_brooklyn = sum_in_Brooklyn.limit(10)
# join với dataframe
result_df = limit_brooklyn.join(df, "neighbourhood", "inner")
# lọc ra quận có số reviews lớn hơn 50
result_df_filtered = result_df.filter(col("number_of_reviews") > 50)
Brooklyn = result_df_filtered.orderBy(col("number_of_reviews").desc())
best_neighbour_in_Brooklyn=Brooklyn.toPandas()
```

```
plt.figure(figsize=(20, 12))
sns.boxplot(x="neighbourhood", y="number_of_reviews", hue="room_type", data=best_neighbour_in_Brooklyn)
plt.show()
```

được gọi là nhím. Nó cũng hiển thị giá trị tối thiểu và tối đa cũng như các ngoại lệ trong dữ liệu.

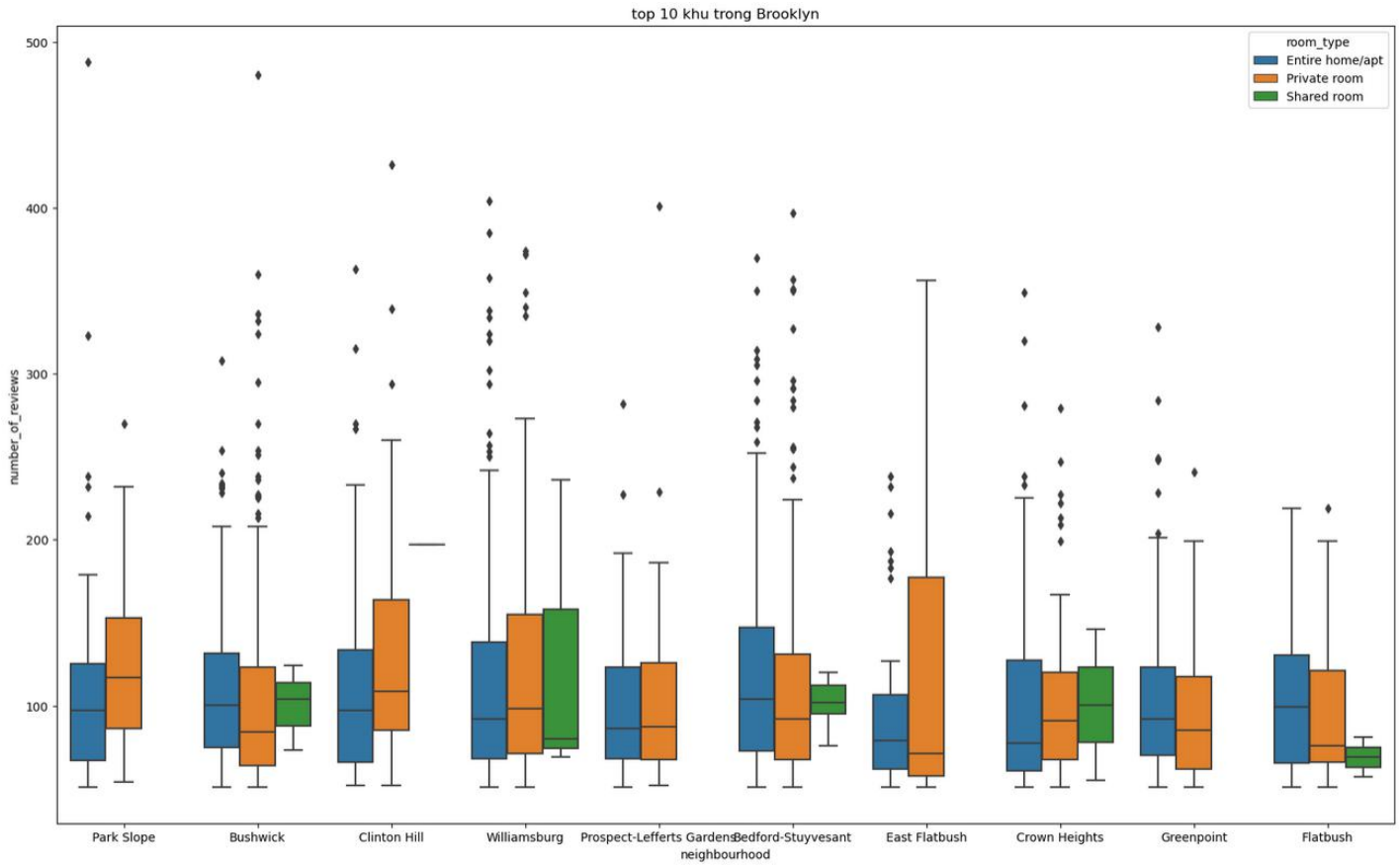
Tính tổng số lượt đánh giá cho từng quận ở Brooklyn và lấy 10 quận có tổng số lượt đánh giá cao nhất.

Lọc ra các quận (neighbourhood) có số lượt đánh giá lớn hơn 50.

Sắp xếp kết quả theo số lượt đánh giá giảm dần.

Vẽ biểu đồ hộp (box plot) bằng Seaborn, với trục x là neighbourhood, trục y là number_of_reviews, và màu sắc được phân loại theo room_type.

Hình minh họa:



Hình 5.4 - Biểu đồ Box plots về Số lượng reviews của top 10 khu trong Brooklyn

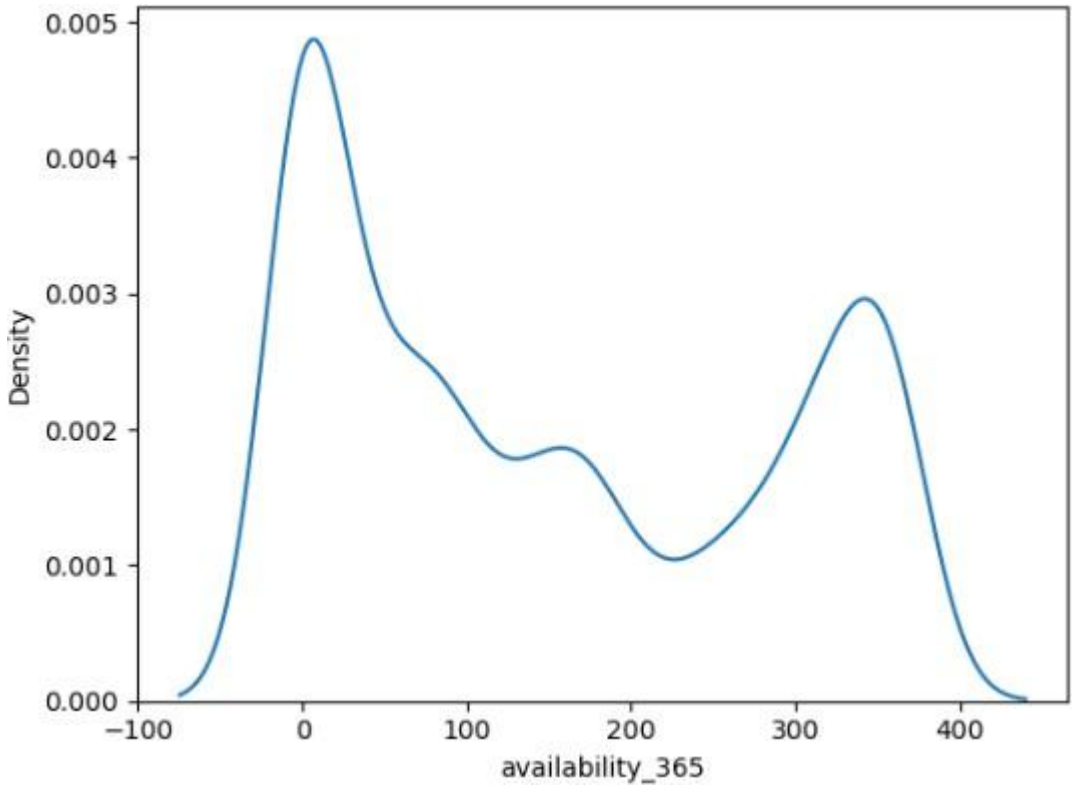
e. KDE plots

Hàm *kde()* vẽ ước lượng mật độ xác suất của một biến số liên tục đã cho. Đó là loại ước lượng không tham số.

```
Queens_neig = df.filter(df['neighbourhood_group']=='Queens')
Queens_neig_pd=Queens_neig.toPandas()
sns.kdeplot(Queens_neig_pd.availability_365)
plt.show()
```

Đầu ra của hàm *kdeplot()* là một đường cong mật độ xác suất ước lượng của biến liên tục availability_365.

Hình minh họa:



Hình 5.5 – Biểu đồ KDE về thống kê số ngày có sẵn của các phòng tại thành phố Queens

f. Violin plots

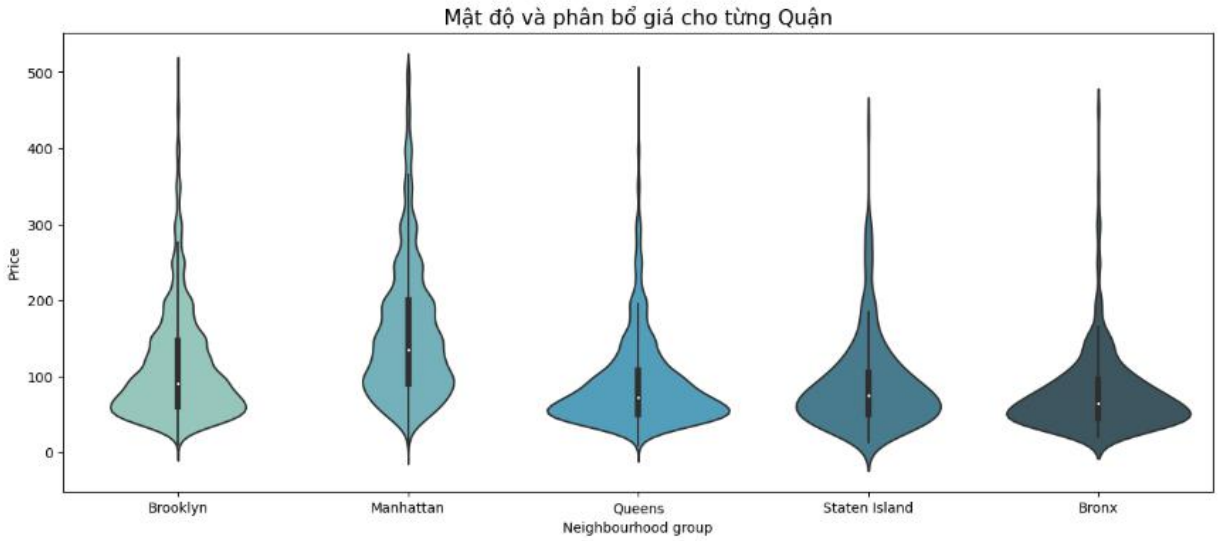
Violin plots là một sự kết hợp của Box plots và KDE, cung cấp phân tích dễ hiểu về phân phối.

```
df_pandas = df.toPandas()
plt.figure(figsize=(15,6))
sns.violinplot(data=df_pandas[df_pandas.price <500], x='neighbourhood_group', y='price', palette='GnBu_d')
plt.title('Mật độ và phân bổ giá cho từng Quận', fontsize=15)
plt.xlabel('Neighbourhood group')
plt.ylabel("Price")
```

Dữ liệu được lọc để chỉ bao gồm các hàng có giá dưới 500.

violinplot() được vẽ với trục x là neighbourhood_group và trục y là price. Màu sắc của biểu đồ được đặt bằng palette = 'GnBu_d'.

Hình minh họa:



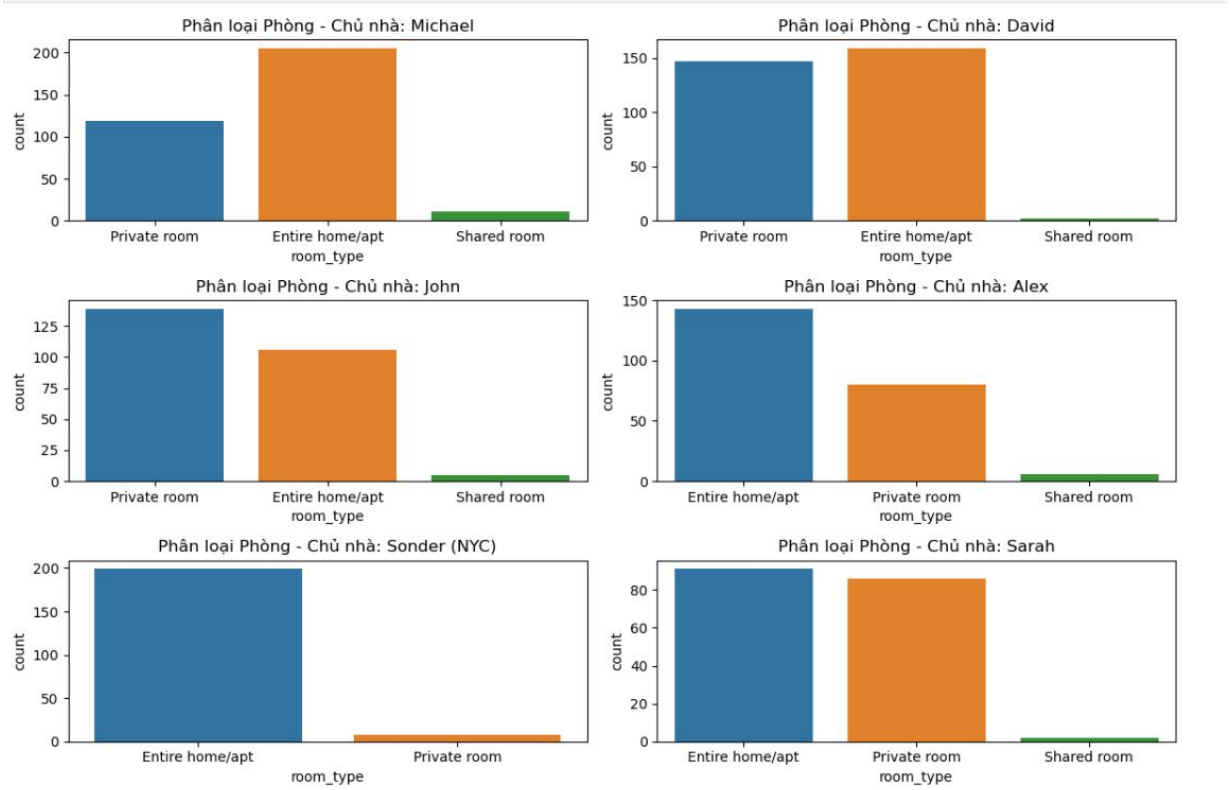
Hình 5.6 – Biểu đồ Violin plots về Mật độ và phân bổ giá cho từng Quận

g. Count plots

Hàm *countplot()* là một loại đặc biệt của biểu đồ cột. Nó hiển thị tần suất của mỗi biến phân loại. Nó còn được biết đến là một loại histogram cho biến phân loại. Nó làm cho các thao tác trở nên đơn giản so với Matplotlib. Và với một chút điều chỉnh, chúng ta có thể sắp xếp các cột theo thứ tự tăng hoặc giảm Hoặc cũng có thể dùng countplot để xem tương quan giữa 2 đặc trưng.

```
df_pandas = df.toPandas()
df_6_host = sum_by_host_sorted_top6.toPandas()

# lấy ra top 6 host_name
top_6_hosts = df_6_host['host_name'].tolist()
# tạo một subplots với 3 hàng 2 cột
fig, axes = plt.subplots(3, 2, figsize=(12, 8))  # figsize điều chỉnh size cho biểu đồ

# lặp qua 6 host để tạo countplot
for i, host_name in enumerate(top_6_hosts):
    host_listings = df_pandas[df_pandas['host_name'] == host_name]

    # tạo countplot
    sns.countplot(x='room_type', data=host_listings, ax=axes[i // 2, i % 2])

    # Set title cho mỗi countplot
    axes[i // 2, i % 2].set_title(f"Phân loại Phòng - Chủ nhà: {host_name}")

# Adjust layout (optional)
plt.tight_layout()  # Adjust spacing between subplots
plt.show()
```

Hình minh họa:



Hình 5.7 – Biểu đồ Count plots về Loại phòng mà 6 Host đang có

Biểu đồ này cho thấy rõ hơn các loại phòng mà top 6 host đang sở hữu :

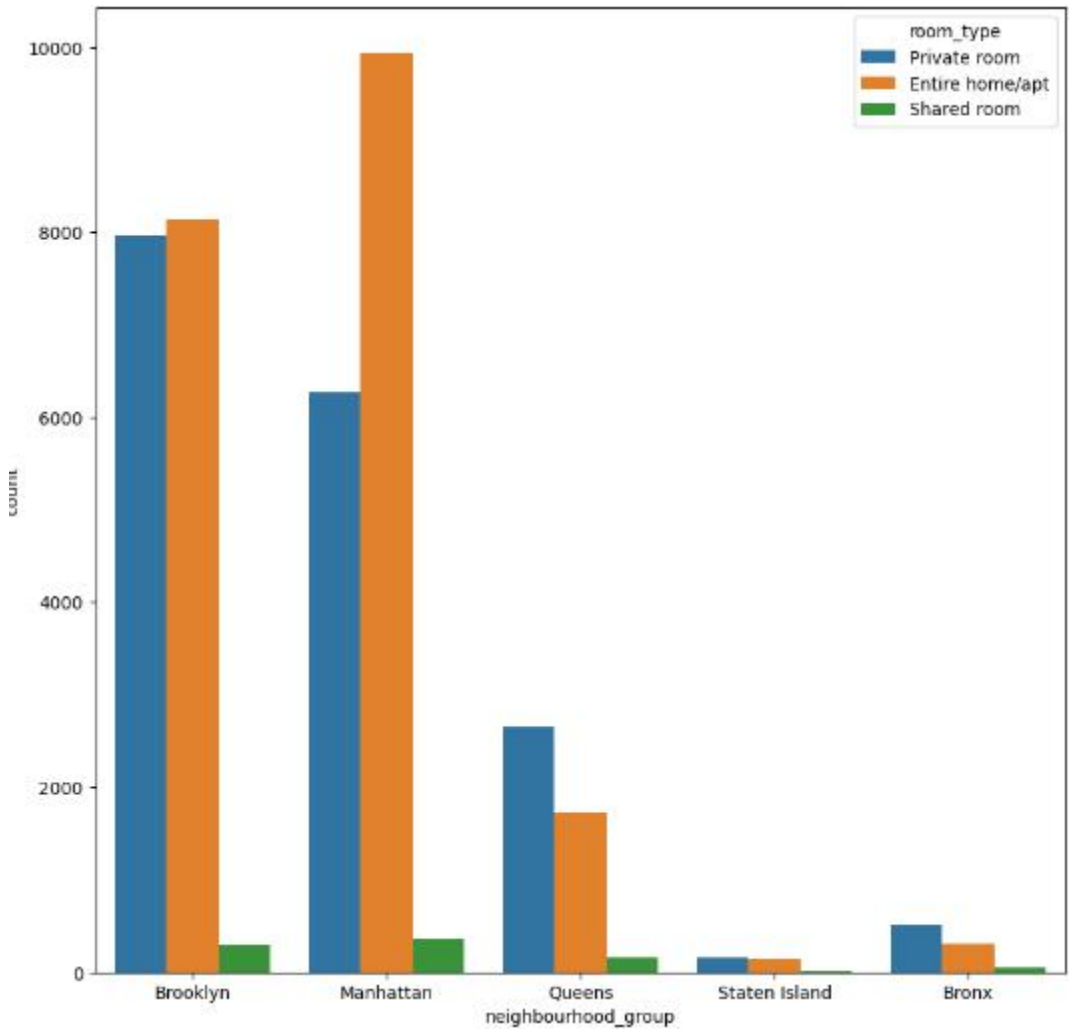
Đa số mọi người sẽ kinh doanh theo Entire home/apt nên số lượng phòng này sẽ chiếm đa số của mỗi chủ sở hữu

Tuy nhiên có Join sẽ kinh doanh nhiều hơn với Private room

Một ví dụ khác về Count plots:

```
df_pandas_loc =df.filter((col('room_type') == "Shared room") | (col('room_type') == "Private room") | (col('room_type') == "Entire home/apt"))
df_pd=df_pandas_loc.toPandas()
plt.figure(figsize=(10,10))
ax = sns.countplot(x=df_pd['neighbourhood_group'],hue=df_pd['room_type'])
```

Hình ảnh minh họa:



Hình 5.8 – Biểu đồ Count plots về Top các quận kinh doanh Airbnb nhiều nhất

Ở đây ta có thể thấy ở Brooklyn và Manhattan có số lượng phòng chênh lệch hơn so với 3 thành phố còn lại

Để ở Entire home/apt thì chúng ta nên tới Manhattan vì nơi đây có nhiều sự lựa chọn hơn so với các thành phố còn lại

Và còn ở Private room thì Brooklyn là sự lựa chọn tốt hơn so với các thành phố còn lại

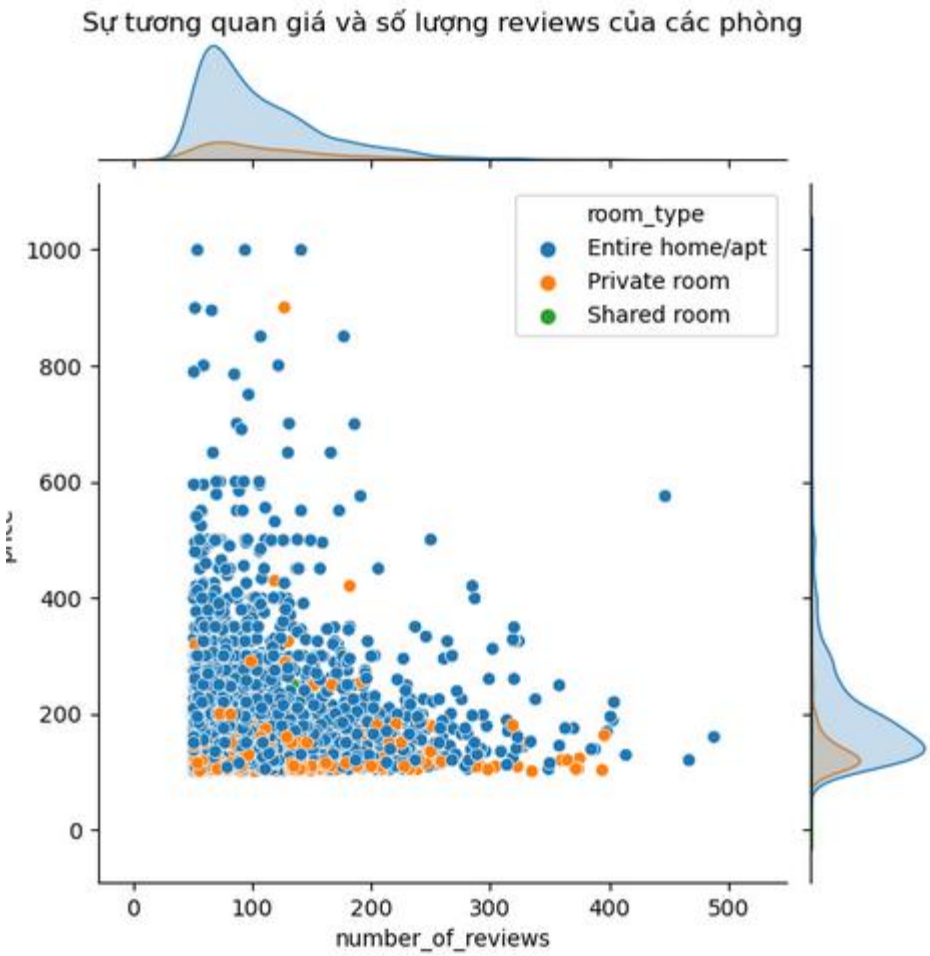
h. Joint plots

Joint plots là một dạng trực quan đa bảng, nó hiển thị mối quan hệ hai chiều và phân phối của các biến số riêng lẻ trong một đồ thị duy nhất. Chúng ta cũng có thể vẽ một KDE bằng cách sử dụng tham số kind của *jointplot()*. Bằng cách đặt tham số kind thành "kde", chúng ta có thể vẽ biểu đồ KDE.

```
# Lọc các dòng có giá lớn hơn 100 và nhỏ hơn 1000 và số lượng reivews > 50
result_df = df.filter((col("price") > 100) & (col("price") < 1000) &(col("number_of_reviews")>50))
# Chuyển kết quả thành Pandas DataFrame
result_df = result_df.toPandas()
```

```
sns.jointplot(data=result_df,x='number_of_reviews', y='price',hue="room_type")
# Show figure
plt.show()
```

Hình minh họa:



Hình 5.9 – Biểu đồ joinplot 'sự tương quan giữa giá và số lượng reivews của các phòng '

Đa số các phòng sẽ tập trung ở mức giá 100 - 600, cũng như có 1 số phòng giá thấp nhưng lượng reviews lớn lên tới 500. Điều đó có nghĩa là không phải phòng nào có giá thấp sẽ có số lượng reviews thấp và loại phòng Entire home/apt chiếm số đông so với các loại còn lại.

i. Heatmaps

Heatmaps được biểu diễn dưới dạng lưới hai chiều. Mỗi ô riêng lẻ của lưới chứa một giá trị của ma trận.

Hàm *corr()* trả về ma trận tương quan. Ma trận tương quan này được vẽ bằng cách sử dụng hàm *heatmap()* để hiển thị lưới của ma trận tương quan.

Chúng ta cũng có thể đặt một colormap mới bằng cách sử dụng tham số *cmap* để có các màu sắc khác nhau. Cũng có thể sử dụng kết hợp màu ví dụ như: YlGnBu (vàng, xanh lá cây và xanh dương) cho *cmap*.

```
# Loại bỏ các cột không phải là số
numeric_df = df_pandas.select_dtypes(include=['number'])

# Tính toán ma trận tương quan
sns.set(font_scale=3)
plt.figure(figsize=(30, 20))
sns.heatmap(numeric_df.corr(), annot=True,fmt=".2f")
```

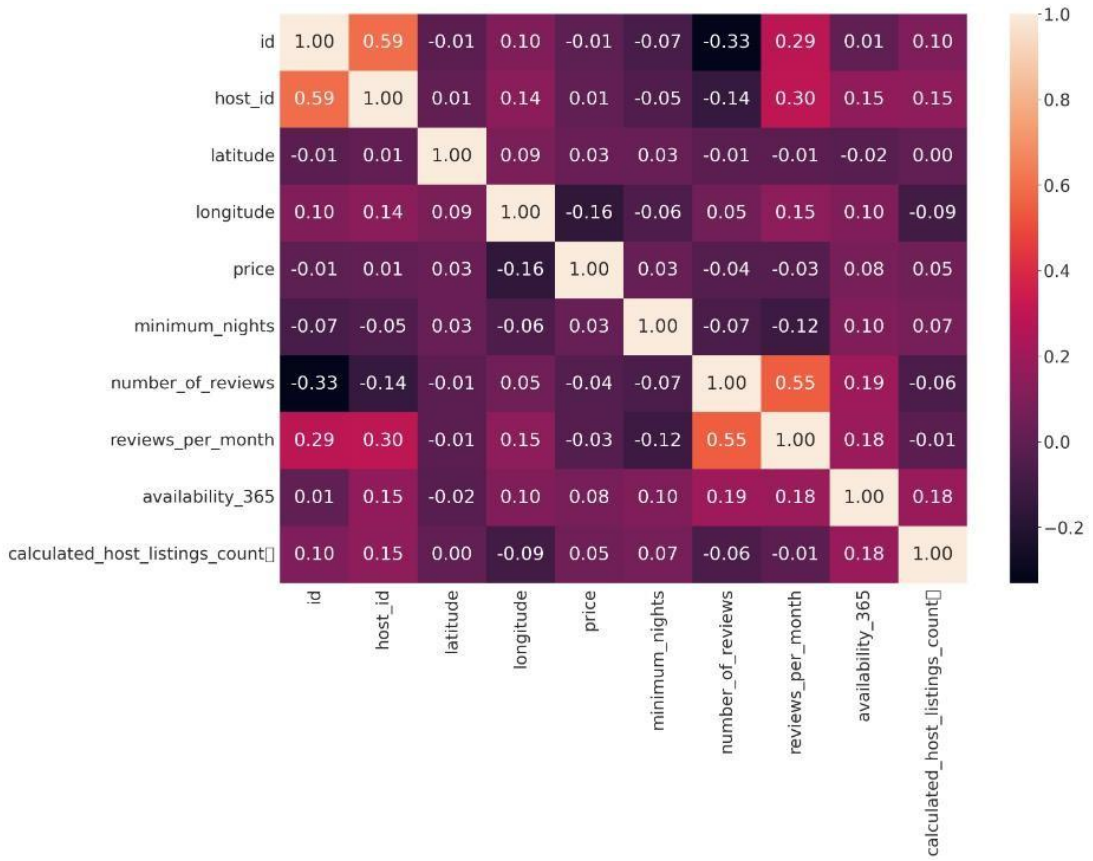
Tính toán ma trận tương quan của các cột số bằng cách sử dụng hàm *corr()*

Các giá trị trong ô được chú thích bằng cách đặt tham số annot=True.

Đặt độ lớn của font cho chú thích bằng cách sử dụng sns.set(font_scale=3).

Đặt kích thước của biểu đồ heatmap bằng cách sử dụng plt.figure(figsize=(30, 20)).

Hình minh họa:



Hình 5.10 – Biểu đồ Heatmaps về sự tương quan của các giá trị AirBnB

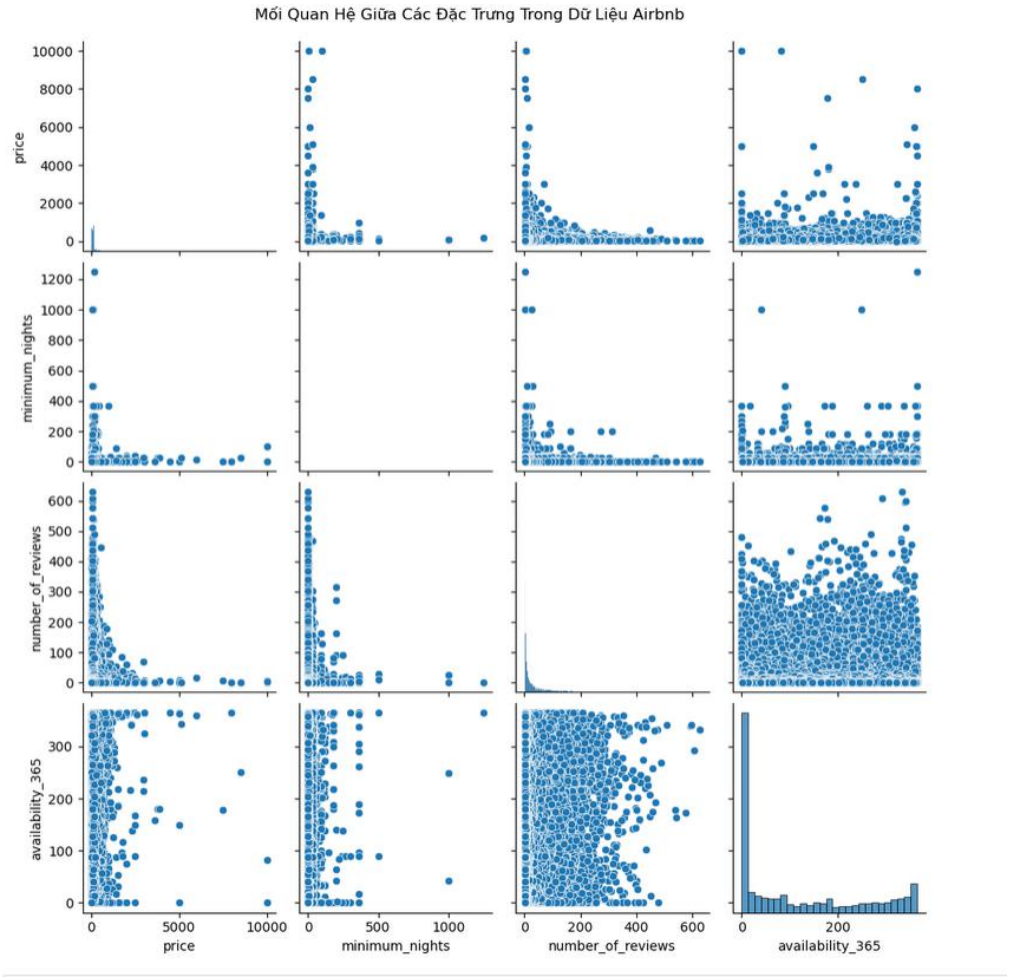
j. Pair plots

Pair plots cung cấp một phân phối đơn lẻ bằng histogram và phân phối chung bằng biểu đồ phân tán. Khi muốn nhìn tổng quan dữ liệu và mối tương quan giữa các chiều dữ liệu theo từng cặp với nhau, nên sử dụng pair plots.

```
selected_df = df.select("price", "minimum_nights", "number_of_reviews", "availability_365").toPandas()
# pair plot
pair_plot = sns.pairplot(selected_df)
# figure chứa các đồ thị của pair plot.
pair_plot.fig.suptitle("Mối Quan Hệ Giữa Các Đặc Trưng Trong Dữ Liệu Airbnb", y=1.02)
plt.show()
```

Ở đoạn code trên, để sử dụng pairplot ta sử dụng *pairplot()* với tham số bên trong là dataframe (selected_df), sử dụng *fig.suptitle()* để set title cho biểu đồ.

Hình minh họa:



Hình 5.11- Biểu đồ Pair plot về mối quan hệ trong Airbnb

- Đường chéo chứa các histogram của mỗi biến.
- Các ô khác chứa scatter plot của cặp biến tương ứng.

6. Bokeh

Bokeh là một thư viện có khả năng tương thích cao với nhiều framework khác, nó có thể dễ dàng làm việc với các ứng dụng trực quan khác nhau, và cả Django. Bạn có thể tùy chỉnh hình ảnh bằng Bokeh. Nó cho phép bạn triển khai các bố cục tương tác và các tính năng đa dạng mô hình trực quan hóa dữ liệu.

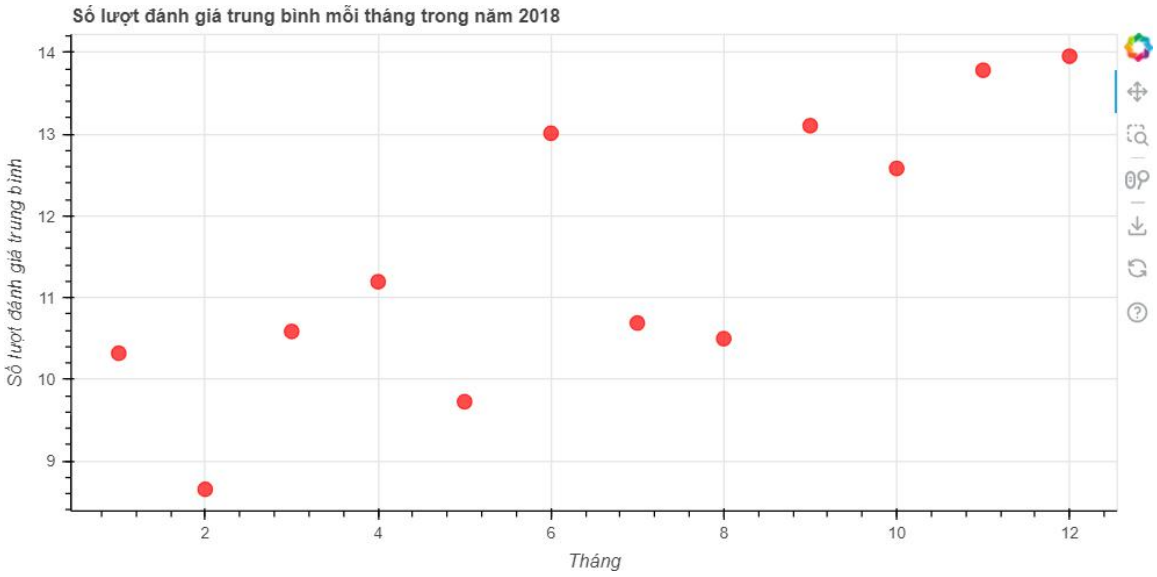
a. Plotting a simple graph

Để có thể hiểu được cách sử dụng Bokeh, ta sẽ vẽ một biểu đồ đơn giản.Trước tiên, chúng ta cần import module bokeh.plotting cơ bản. Hàm output_notebook() xác định rằng biểu đồ sẽ được hiển thị trên Jupyter Notebook. Đối tượng figure được sử dụng như một trong những đối tượng cốt lõi để vẽ biểu đồ và đồ thị. Đối tượng figure tập trung vào tiêu đề biểu đồ, kích thước, nhãn, lưới và kiểu dáng. Đối tượng figure cũng xử lý kiểu dáng biểu đồ, tiêu đề, nhãn trục, trục, lưới và các phương thức khác để thêm dữ liệu.

```
# Lọc dữ liệu cho năm 2018
airbnb_2018 = df.filter(year(df['last_review']) == 2018)

# Tính số lượng đánh giá trung bình theo tháng
reviews_per_month_2018 = airbnb_2018.groupBy(month(airbnb_2018['last_review']).alias('month')) \
                                    .agg(avg('number_of_reviews').alias('avg_reviews')).orderBy('month')

# Output vào notebook
output_notebook()

# Tạo ColumnDataSource từ dữ liệu
source = ColumnDataSource(reviews_per_month_2018.toPandas())

# Tạo figure object
fig = figure(width=800, height=400, title="Số lượt đánh giá trung bình mỗi tháng trong năm 2018",
             x_axis_label='Tháng', y_axis_label='Số lượt đánh giá trung bình')

# Vẽ scatter plot bằng cách sử dụng ColumnDataSource
fig.circle(x='month', y='avg_reviews', size=10, color="red", alpha=0.7, source=source)

# Hiển thị biểu đồ
show(fig)
```

Trong đoạn code trên, sử dụng figure() để miêu tả độ dài,cao cũng như là tiêu đề của biểu đồ, tiêu đề trục x,y.

Hình ảnh minh họa :



Hình 6.1 – Biểu đồ 'trung bình số lượt đánh giá trung bình mỗi tháng trong 2018'

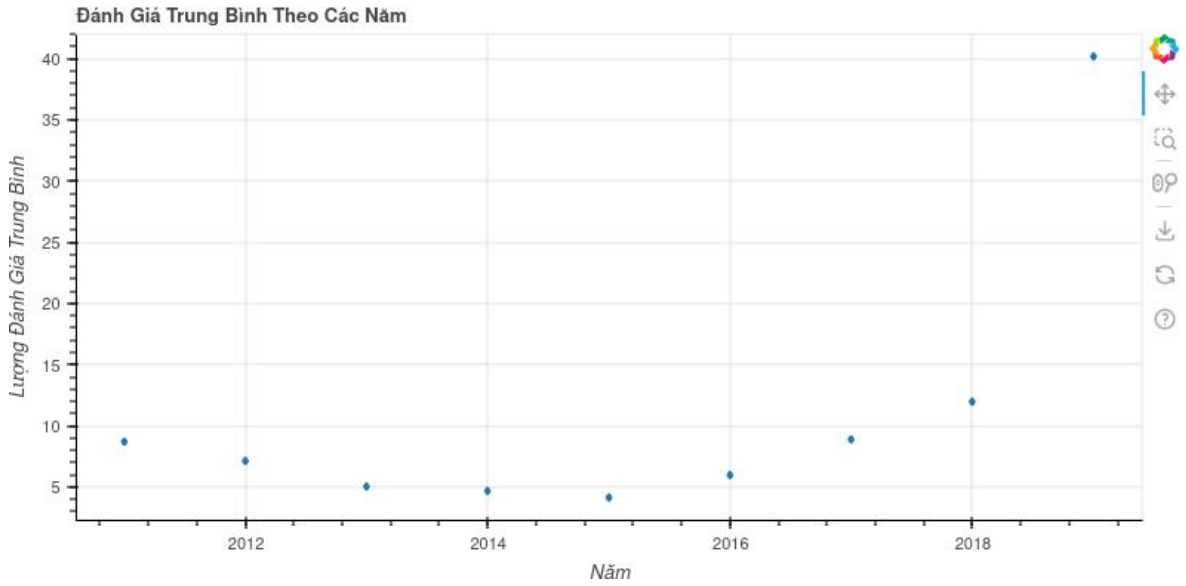
b. Glyphs

Bokeh sử dụng visual glyph, đề cập đến các hình tròn(circles), đường thẳng(lines), tam giác(triangles), vuông(squares), thanh(bars), kim cương(diamonds) và các đồ thị hình dạng khác. Glyph là một biểu tượng duy nhất được sử dụng để truyền đạt thông tin dưới dạng hình ảnh.

```
# Tạo ColumnDataSource từ Pandas DataFrame
# average_reviews_by_year_pd là dataframe lấy theo reviews trung bình của năm
source = ColumnDataSource(average_reviews_by_year_pd)

# Tạo figure object
p = figure(width=800, height=400, title="Đánh Giá Trung Bình Theo Các Năm",
           x_axis_label='Năm', y_axis_label='Lượng Đánh Giá Trung Bình')

# Vẽ biểu đồ diamond sử dụng glyph line
p.diamond(x='year', y='avg(number_of_reviews)', source=source, line_width=2)

# Hiển thị biểu đồ
show(p)
```

Sử dụng diamond() để thiết lập biểu đồ vẽ kim cương với hàng x là năm và y là số đánh giá trung bình, source cung cấp dữ liệu cho diamond và line_width là độ dày.

Hình ảnh minh họa:



Hình 6.2 – Biểu đồ 'Trung bình đánh theo từng năm'

c. Layouts

Bokeh cung cấp các Layouts để sắp xếp các biểu đồ và các tiện ích. Các Layouts này giúp sắp xếp nhiều biểu đồ trong một bảng đơn cho các trực quan hóa tương tác. Chúng cũng cho phép thiết lập các chế độ kích thước để tự động điều chỉnh kích thước của các biểu đồ và tiện ích dựa trên kích thước của bảng.

Có các loại bố cục sau:

Bố cục dạng dòng(Row layout): Sắp xếp tất cả các biểu đồ theo hàng hoặc theo chiều ngang.

Bố cục dạng cột(Column layout): Sắp xếp tất cả các biểu đồ theo cột hoặc theo chiều dọc.

Bố cục lồng nhau(Nested layout): Kết hợp bố cục dạng dòng và dạng cột.

Bố cục lưới(Grid layout): Cung cấp một lưới các ma trận để sắp xếp các biểu đồ.

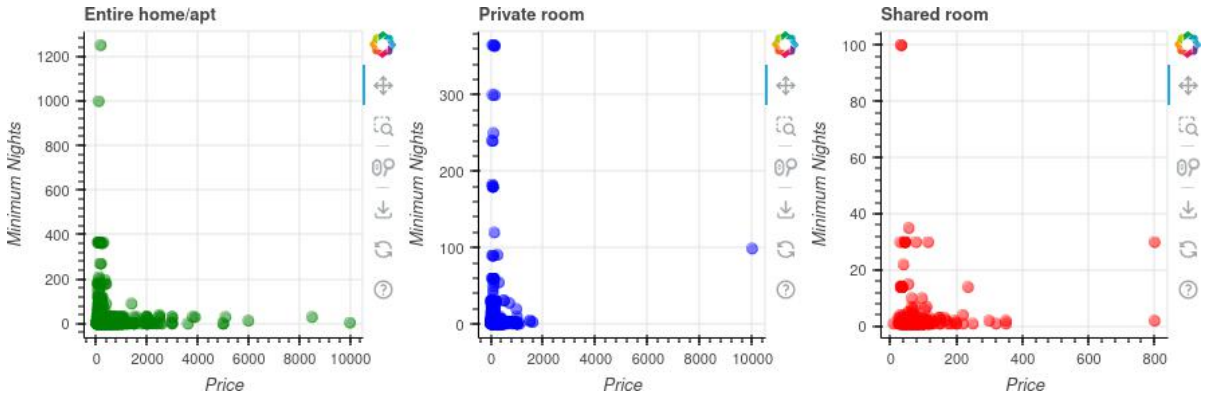
Các bố cục này cho phép tổ chức và hiển thị các biểu đồ một cách hiệu quả, giúp tăng cường khả năng trình bày thông tin trong các ứng dụng trực tuyến và tương tác.

Để có thể dễ hiểu hơn thì ví dụ về một row layout :

```
# Lọc dữ liệu cho các loại phòngthuộc khu vực Manhattan
df_entire_home = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Entire home/apt')).select('price', 'minimum_nights').toPandas()
df_private_room = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Private room')).select('price', 'minimum_nights').toPandas()
df_shared_room = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Shared room')).select('price', 'minimum_nights').toPandas()

# Output vào notebook
output_notebook()

# Khởi tạo các đối tượng figure cho từng subplot
fig_entire_home = figure(width=300, height=300, title="Entire home/apt",x_axis_label='Price', y_axis_label='Minimum Nights')
fig_private_room = figure(width=300, height=300, title="Private room",x_axis_label='Price', y_axis_label='Minimum Nights')
fig_shared_room = figure(width=300, height=300, title="Shared room",x_axis_label='Price', y_axis_label='Minimum Nights')

# Tạo biểu đồ scatter marker cho loại phòng
fig_entire_home.circle(df_entire_home['price'], df_entire_home['minimum_nights'], size=8, color="green", alpha=0.5)
fig_private_room.circle(df_private_room['price'], df_private_room['minimum_nights'], size=8, color="blue", alpha=0.5)
fig_shared_room.circle(df_shared_room['price'], df_shared_room['minimum_nights'], size=8, color="red", alpha=0.5)

# Tạo một row layout
row_layout = row(fig_entire_home, fig_private_room, fig_shared_room)


# Hiển thị biểu đồ
show(row_layout)
```

Ở đoạn code trên ta sử dụng *row()* và truyền các tham số là các biểu đồ mình đã tạo để cho các biểu đồ được đặt theo chiều ngang.

Hình ảnh minh họa:



Hình 6.3 – Biểu đồ 'Số lượng các loại phòng có ở Manhattan' theo hàng

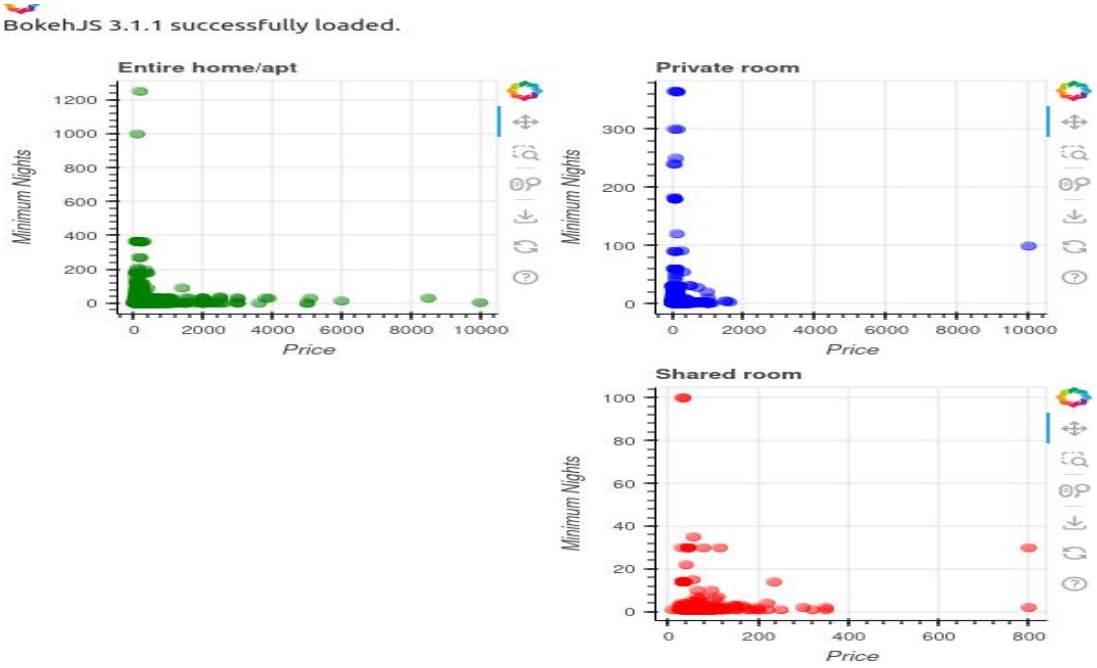
Nested layout using row and column layouts:

Ví dụ về một biểu đồ vừa hàng vừa cột :

```
## NESTED lAYOUT USING ROW AND COLUMN LAYOUTS

df_entire_home = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Entire home/apt')).select('price', 'minimum_nights').toPandas()
df_private_room = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Private room')).select('price', 'minimum_nights').toPandas()
df_shared_room = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Shared room')).select('price', 'minimum_nights').toPandas()

# Output vào notebook
output_notebook()

# Khởi tạo các đối tượng figure cho từng subplot
fig_entire_home = figure(width=300, height=300, title="Entire home/apt",x_axis_label='Price', y_axis_label='Minimum Nights')
fig_private_room = figure(width=300, height=300, title="Private room",x_axis_label='Price', y_axis_label='Minimum Nights')
fig_shared_room = figure(width=300, height=300, title="Shared room",x_axis_label='Price', y_axis_label='Minimum Nights')

# Tạo biểu đồ scatter marker cho loại phòng Toàn bộ nhà/căn hộ
fig_entire_home.circle(df_entire_home['price'], df_entire_home['minimum_nights'], size=8, color="green", alpha=0.5)

# Tạo biểu đồ scatter marker cho loại phòng Phòng riêng
fig_private_room.circle(df_private_room['price'], df_private_room['minimum_nights'], size=8, color="blue", alpha=0.5)

# Tạo biểu đồ scatter marker cho loại phòng Phòng chia sẻ
fig_shared_room.circle(df_shared_room['price'], df_shared_room['minimum_nights'], size=8, color="red", alpha=0.5)

# Tạo một nested layout
nested_layout = row(fig_entire_home, column(fig_private_room, fig_shared_room))

# Hiển thị biểu đồ
show(nested_layout)
```

Trong đoạn code này, sau khi có được ba biểu đồ, chúng ta sử dụng *row(column())* để set biểu đồ theo chiều ngang và dọc, ở đây sẽ có biểu đồ fig_entire_home sẽ ở hàng 1, và giá trị có trong column sẽ phân bổ theo chiều dọc trên hàng 1 đó.

Hình minh họa :



Hình 6.3.1 - Biểu đồ 'Số lượng các loại phòng có ở Manhattan' theo cột và hàng

d. Multiple plots

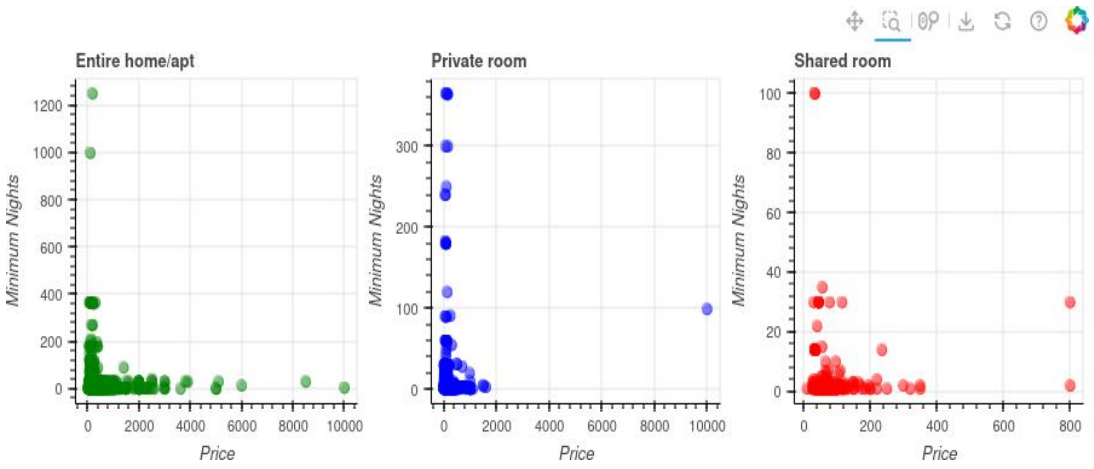
Có thể tạo nhiều biểu đồ và đối tượng sử dụng một grid layout. Grid layout sắp xếp các biểu đồ và đối tượng widget theo kiểu ma trận hàng-cột. Nó nhận một danh sách các đối tượng figure cho mỗi hàng. Nó hiển thị và giúp mình so sánh dễ hơn nhiều biểu đồ một cách hiệu quả.

```
## MULTIPLE PLOTS

df_entire_home = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Entire home/apt')).select('price', 'minimum_nights').toPandas()
df_private_room = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Private room')).select('price', 'minimum_nights').toPandas()
df_shared_room = df.filter((df['neighbourhood_group'] == 'Manhattan') & (df['room_type'] == 'Shared room')).select('price', 'minimum_nights').toPandas()

# Output vào notebook
output_notebook()
# Khởi tạo các đối tượng figure cho từng subplot
fig_entire_home = figure(width=300, height=300, title="Entire home/apt",x_axis_label='Price', y_axis_label='Minimum Nights')
fig_private_room = figure(width=300, height=300, title="Private room",x_axis_label='Price', y_axis_label='Minimum Nights')
fig_shared_room = figure(width=300, height=300, title="Shared room",x_axis_label='Price', y_axis_label='Minimum Nights')
# Tạo biểu đồ scatter marker cho loại phòng
fig_entire_home.circle(df_entire_home['price'], df_entire_home['minimum_nights'], size=8, color="green", alpha=0.5)


fig_private_room.circle(df_private_room['price'], df_private_room['minimum_nights'], size=8, color="blue", alpha=0.5)

fig_shared_room.circle(df_shared_room['price'], df_shared_room['minimum_nights'], size=8, color="red", alpha=0.5)

# Tạo một grid layout
grid_layout = gridplot([[fig_entire_home, fig_private_room, fig_shared_room]])

# Hiển thị biểu đồ
show(grid_layout)
```

Trong đoạn code trên, sau khi tạo ra các biểu đồ của từng loại phòng, chúng ta sử dụng *gridplot()*, *gridplot()* là một bố cục lưới cho các biểu đồ, lúc này *gridplot()* sẽ lấy danh sách 2D của các đối tượng biểu đồ, và các tham số khác như :

i. Ncols : sẽ lấy giá trị mặc định dựa trên số lượng con (lúc này là 3)
ii. Width,height: sẽ lấy giá trị mặc định của mỗi đối tượng của biểu đồ
iii. Toolbar_location: sẽ lấy giá trị mặc định là ở trên biểu đồ.

Hình ảnh minh họa :



Hình 6.4 - Biểu đồ gridplot 'Số lượng các loại phòng có ở Manhattan'

e. Interactions

Trong Bokeh, chúng ta có thể tạo các legends cho các biểu đồ để giải thích ý nghĩa của các đối tượng trên biểu đồ, chẳng hạn như các dòng, điểm, hoặc cột. Legends cho phép người xem hiểu được mối quan hệ giữa các yếu tố trên biểu đồ một cách dễ dàng. Legends có thể được ẩn hoặc làm mờ bằng cách nhấp vào các mốc trên biểu đồ. Chúng ta có thể kích hoạt các chế độ này bằng cách kích hoạt thuộc tính click_policy và nhấp vào mục trong huyền thoại.

Có 2 loại :

Click ẩn (Hide click policy) và Click mờ(Mute click policy)

Hide click policy

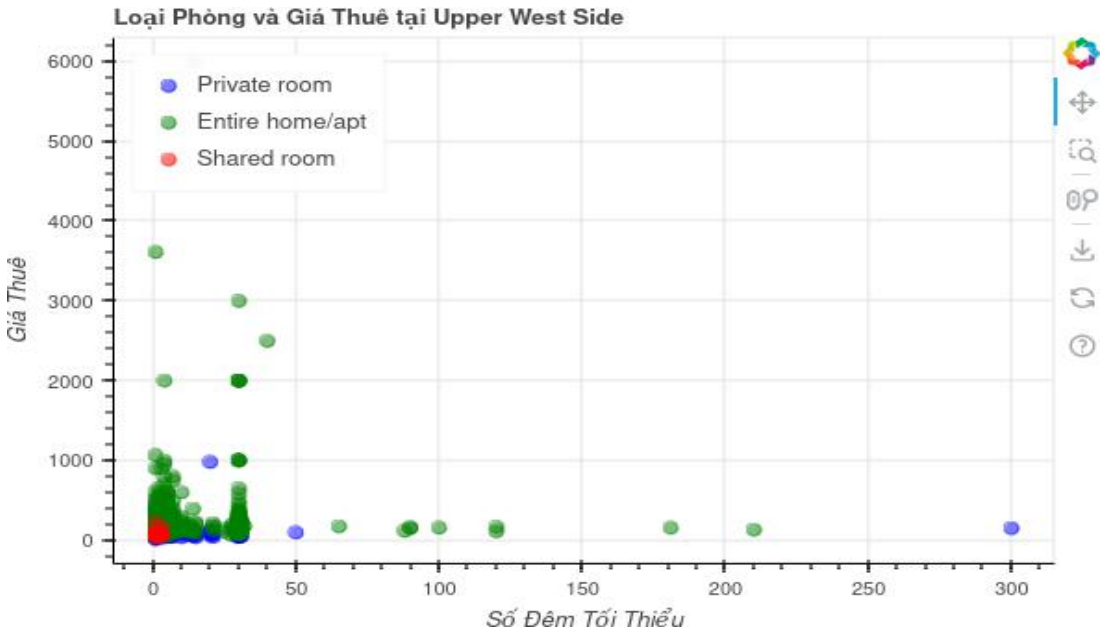
Click ẩn đi các hình tượng mong muốn trong mục legend. Này có thể cho mình lọc ra được thứ mình cần xem.Dưới đây là code để biểu hiện sự hide click policy:

```
## HIDE CLICK POLICY
# Lọc dữ liệu cho các căn hộ thuộc Manhattan và chọn các cột 'room_type', 'price', 'minimum_nights'
df_upper = df.filter(df['neighbourhood'] == 'Upper West Side').select('room_type', 'price','minimum_nights').toPandas()

# Output vào notebook
output_notebook()

# Tạo đối tượng figure
fig = figure(width=600, height=400, title="Loại Phòng và Giá Thuê tại Upper West Side",
           x_axis_label='Số Đêm Tối Thiểu', y_axis_label='Giá Thuê')

# Vòng lặp để tạo các điểm dữ liệu dựa trên các cột trong DataFrame
for room_type, color in zip(df_upper['room_type'].unique(), ['blue', 'green', 'red']):
    data = df_manhattan[df_upper['room_type'] == room_type]
    fig.circle('minimum_nights', 'price', size=8, color=color, alpha=0.5, legend_label=room_type, source=ColumnDataSource(data))

# Đặt vị trí của chú thích
fig.legend.location = 'top_left'
fig.legend.click_policy = "hide"

# Hiển thị biểu đồ
show(fig)
```

Đoạn code trên cho thấy sau khi tạo biểu đồ, để có thể tương tác được trên mục legend mình sẽ sử dụng *fig.legend.click_policy=”hide”* tham số “hide” cho biết legend này sẽ cho mình có thể hide click ẩn đi giá trị trong biểu đồ .

Hình minh họa :

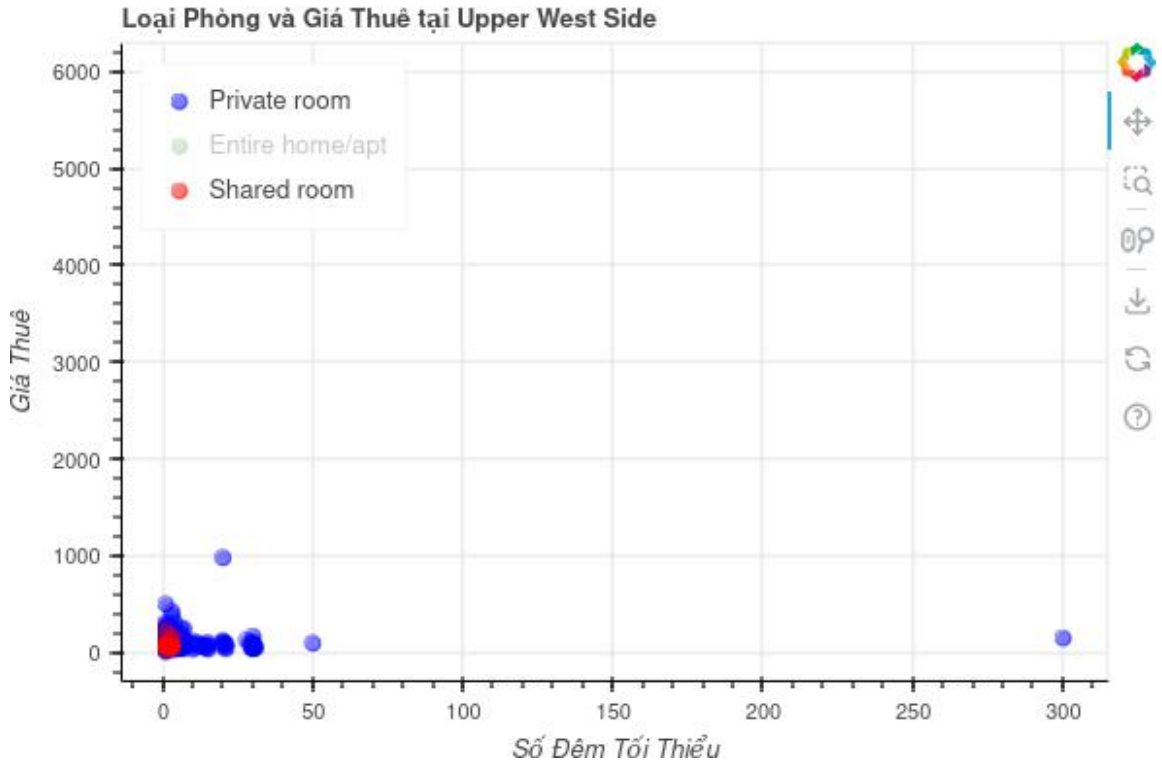
Trước khi hide click :



Hình 6.5 - Biểu đồ 'Loại Phòng và Giá thuê ở upper west side' trước khi hide click

Hình minh họa :

Sau khi hide click:



Hình 6.5.1 - Biểu đồ 'Loại Phòng và Giá thuê ở upper west side'sau khi hide click

2 biểu đồ có sự khác nhau, sau khi ấn ở mục Legend để ẩn đi Entire home/apt thì biểu đồ 2 không còn thấy được giá trị của phòng đó nữa .

Mute click policy

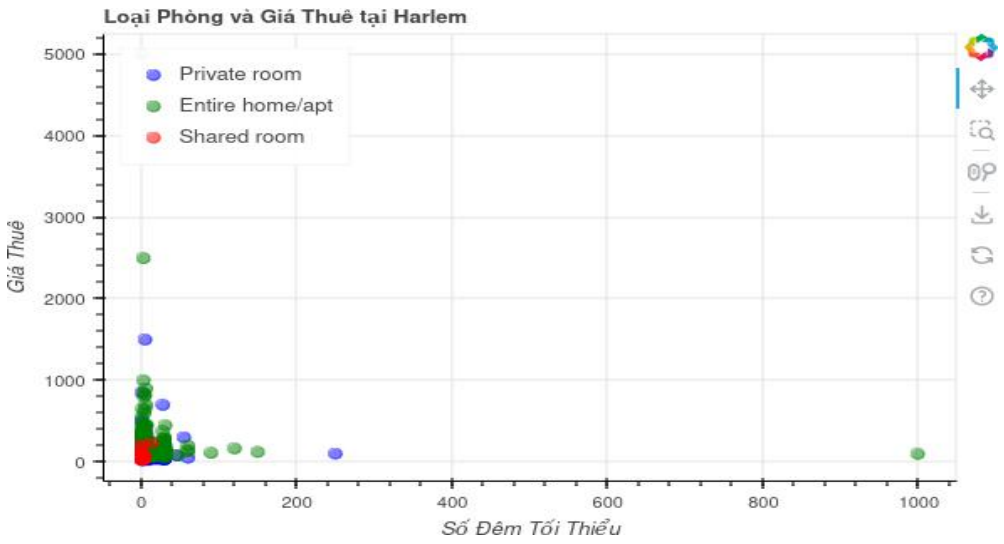
Khác với Hide Click policy thì Mute click policy (Click mờ) sẽ làm mờ đi các hình tượng mong muốn trong mục legend0.Dưới đây là code để biểu hiện sự mute click policy :

```
## MUTE CLICK POLICY

# Lọc dữ liệu cho các căn hộ thuộc Manhattan và chọn các cột 'room_type', 'price', 'minimum_nights'
df_Harlem = df.filter(df['neighbourhood'] == 'Harlem').select('room_type', 'price','minimum_nights').toPandas()

# Output vào notebook
output_notebook()

# Tạo đối tượng figure
fig = figure(width=600, height=400, title="Loại Phòng và Giá Thuê tại Harlem",
           x_axis_label='Số Đêm Tối Thiểu', y_axis_label='Giá Thuê')

# Vòng lặp để tạo các điểm dữ liệu dựa trên các cột trong DataFrame
for room_type, color in zip(df_Harlem['room_type'].unique(), ['blue', 'green', 'red']):
    data = df_Harlem[df_Harlem['room_type'] == room_type]
    fig.circle('minimum_nights', 'price', size=8, color=color, alpha=0.5, legend_label=room_type, source=ColumnDataSource(data))

# Đặt vị trí của chú thích
fig.legend.location = 'top_left'
fig.legend.click_policy = "mute"

# Hiển thị biểu đồ
show(fig)
```

Đoạn code trên cho thấy sau khi tạo biểu đồ, để có thể tương tác được trên mục legend mình sẽ sử dụng *fig.legend.click_policy=”mute”* tham số “mute” cho biết legend này sẽ cho mình có thể mute click mờ đi giá trị trong biểu đồ .

Hình minh họa :

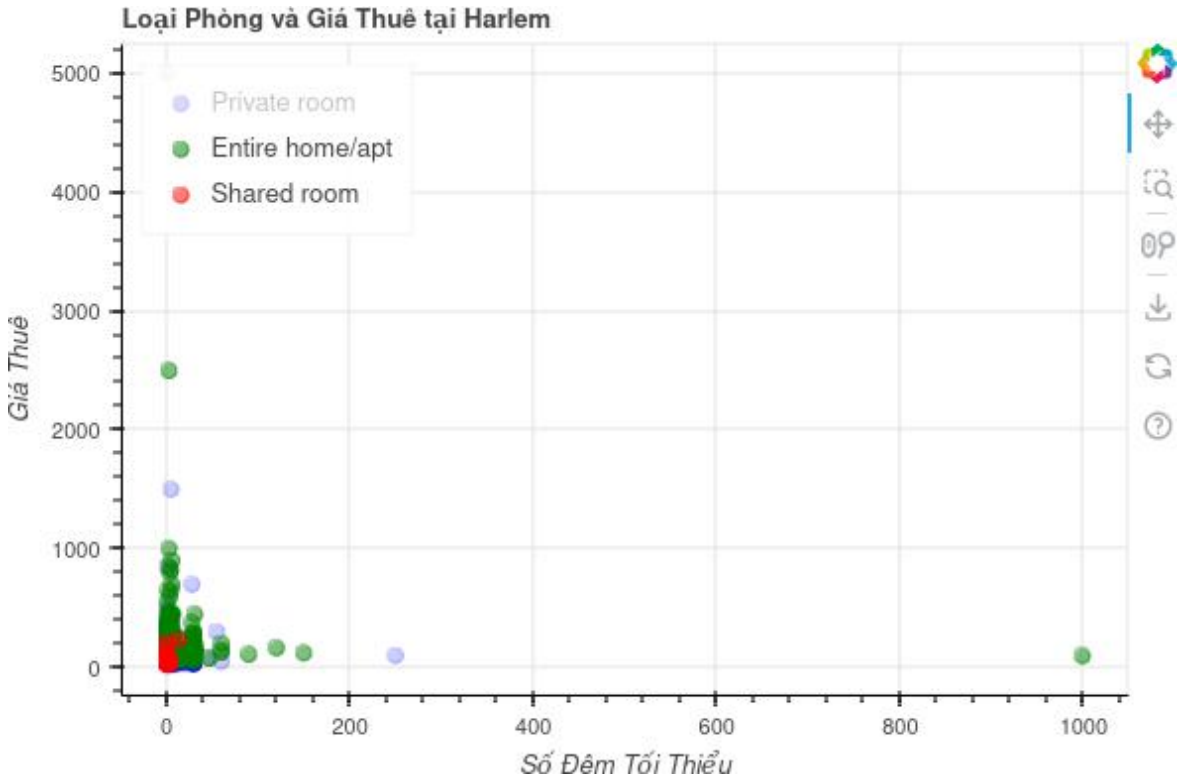
Trước khi mute click :



Hình 6.5.2 - Biểu đồ ‘Loại Phòng và Giá thuê ở Harlem’ trước khi mute click

Hình minh họa :

Sau khi mute click :



Hình 6.5.3 - Biểu đồ 'Loại Phòng và Giá thuê ở Harlem' sau khi mute click

2 biểu đồ có sự khác nhau ở chỗ, sau khi ấn ở mục Legend để mờ đi Share room, thì biểu đồ 6.5.3 thấy được giá trị của phòng đó mờ đi.

f. Annotations

Chú thích trong Bokeh giúp cho thông tin được bổ sung để dễ hiểu hơn về biểu đồ, có một số chú thích gồm :

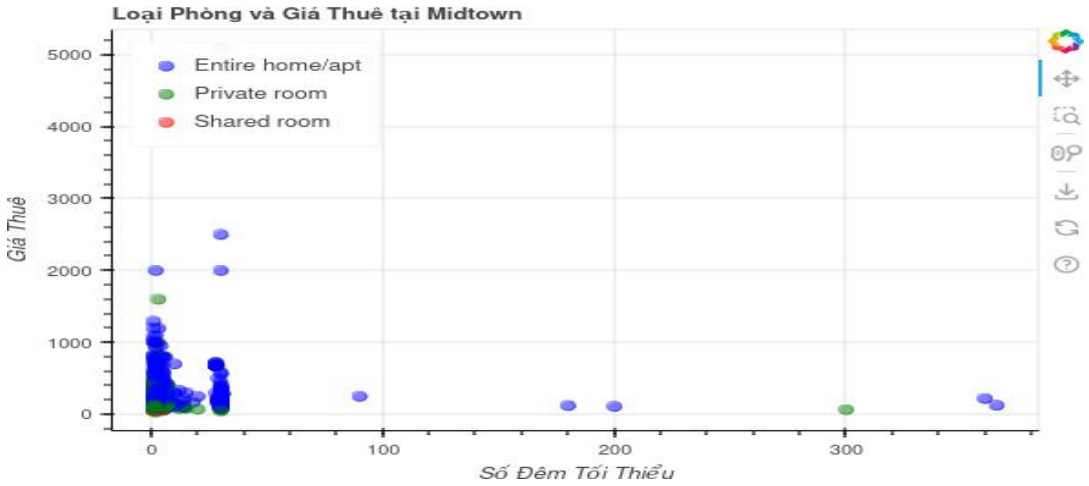
i. Tiêu đề(Tittle): Tên biểu đồ
ii. Nhãn trục(Axis labels): cung cấp nhãn cho trục x,y

iii. Chú thích(Legends) : đại diện cho biến thứ ba thông qua màu sắc hay hình dạng
iv. Color bars: Thanh màu được tạo bằng ColorMapper với bảng màu

```
## ANNOTATIONS

# Lọc dữ liệu cho các căn hộ thuộc Manhattan và chọn các cột 'room_type', 'price', 'minimum_nights'
df_Midtown = df.filter(df['neighbourhood'] == 'Midtown').select('room_type', 'price','minimum_nights').toPandas()

# Tạo một ColumnDataSource từ DataFrame pandas
source = ColumnDataSource(df_Midtown)

# Output vào notebook
output_notebook()

# Tạo color mapper cho cột phân loại
mapper = CategoricalColorMapper(factors=['Entire home/apt', 'Private room', 'Shared room'],
                                palette=['blue', 'green', 'red'])
color_dict = {'field': 'room_type', 'transform': mapper}

# Tạo đối tượng figure
p = figure(width=600, height=400, title="Loại Phòng và Giá Thuê tại Midtown",
           x_axis_label='Số Đêm Tối Thiểu', y_axis_label='Giá Thuê')

# Tạo biểu đồ scatter plot bằng cách vẽ các hình tròn
p.circle('minimum_nights', 'price', size=8, color=color_dict, alpha=0.5, legend_group='room_type', source=source)

# Đặt vị trí của chú thích
p.legend.location = 'top_left'

# Hiển thị biểu đồ
show(p)
```

Ở đoạn code trên ta có thể thấy được một số annotations như *title* để đặt tên cho biểu đồ, *x_axis_label,y_axis_label* 2 giá trị này là tên của hàng x và cột y, mapper sử dụng ColorMapper cho cột phân loại,*legend_group* là nhóm *room_type* để chú thích các phòng .

Hình ảnh minh họa :



Hình 6.6 – Biểu đồ 'loại phòng và giá thuê tại Midtown'

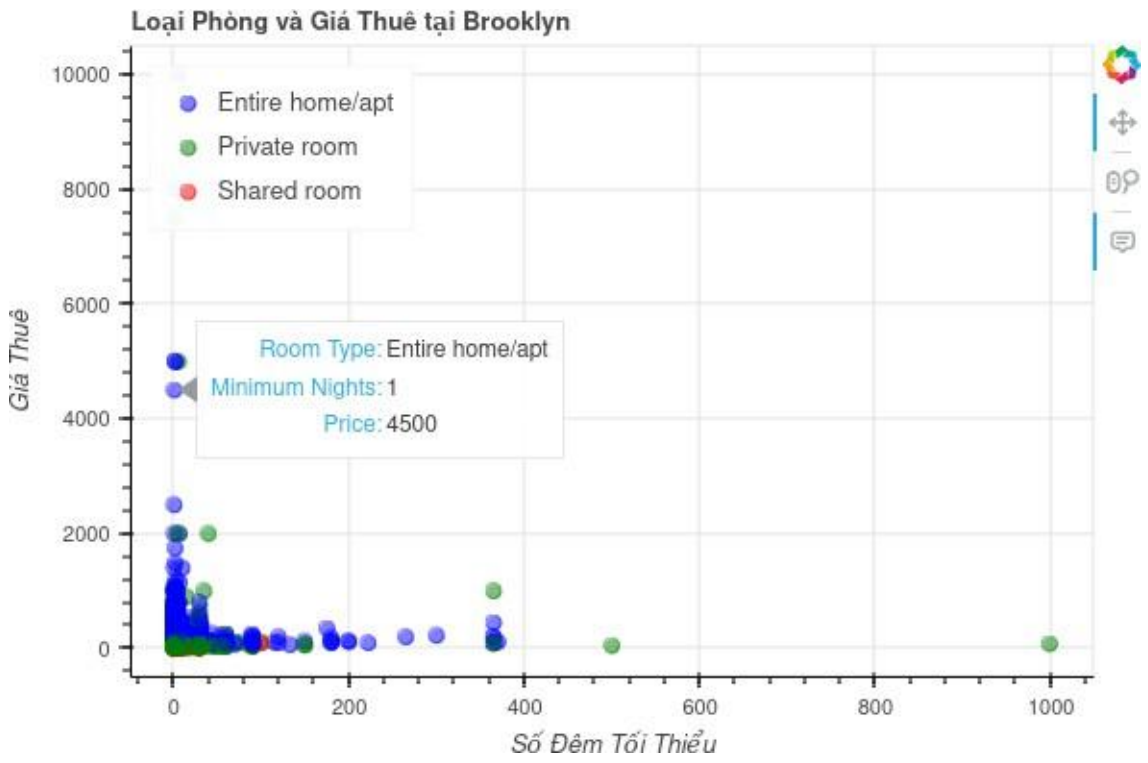
g. Hover tool

Công cụ di chuột (hover tool) hiển thị thông tin liên quan khi con trỏ chuột được đặt qua một vùng cụ thể.

```
## HOVEL TOOL


# Lọc dữ liệu cho các căn hộ thuộc Manhattan và chọn các cột 'room_type', 'price', 'minimum_nights'
df_Brooklyn= df.filter(df['neighbourhood_group'] == 'Brooklyn').select('room_type', 'price','minimum_nights').toPandas()

# Tạo một ColumnDataSource từ DataFrame pandas
source = ColumnDataSource(df_Brooklyn)

# Output vào notebook
output_notebook()

# Tạo color mapper cho cột phân loại
mapper = CategoricalColorMapper(factors=['Entire home/apt', 'Private room', 'Shared room'],
                                palette=['blue', 'green', 'red'])
color_dict = {'field': 'room_type', 'transform': mapper}

# Tạo hover tool và chỉ định thông tin khi di chuột qua
hover = HoverTool(tooltips=[('Room Type', '@room_type'),
                            ('Minimum Nights', '@minimum_nights'),
                            ('Price', '@price')])

# Tạo đối tượng figure
p = figure(width=600, height=400, title="Loại Phòng và Giá Thuê tại Brooklyn",
           x_axis_label='Số Đêm Tối Thiểu', y_axis_label='Giá Thuê',
           tools=[hover, 'pan', 'wheel_zoom'])

# Tạo biểu đồ scatter plot bằng cách vẽ các hình tròn
p.circle('minimum_nights', 'price', size=8, color=color_dict, alpha=0.5, legend_group='room_type', source=source)

# Đặt vị trí của chú thích
p.legend.location = 'top_left'

# Hiển thị biểu đồ
show(p)
```

Như ở đoạn code trên, để có thể khởi tạo được hover, chúng ta sẽ sử dụng *HoverTool()* tooltips ở bên trong là những giá trị mình muốn hiện ra khi mình di chuột đến.

Hình minh họa:



Hình 6.7 – Biểu đồ 'loại phòng và giá thuê tại Brooklyn" (Hover tool)

h. Widgets

Tab panel

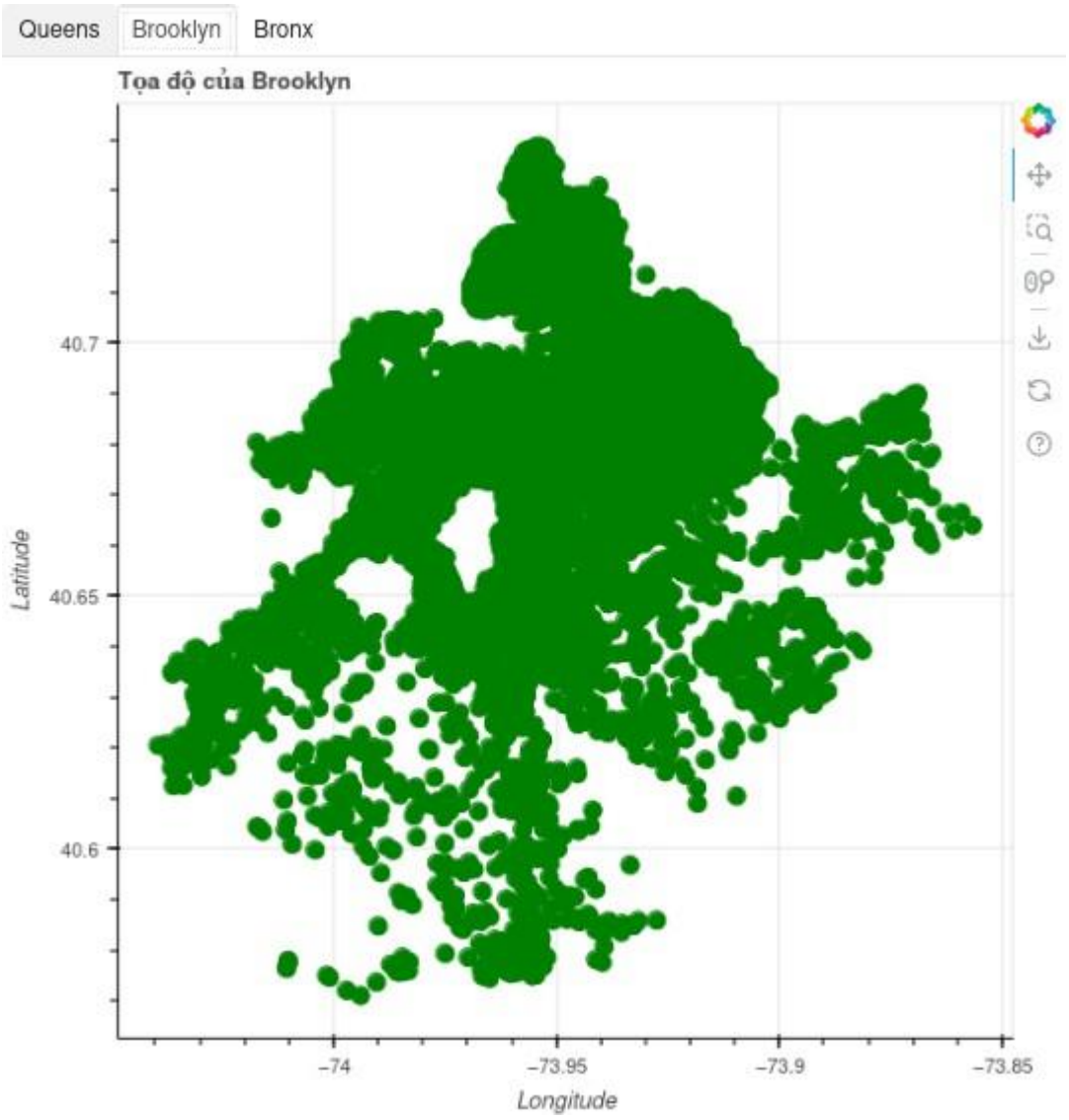
Các tab panes cho phép chúng ta tạo nhiều biểu đồ và bố cục trong một cửa sổ duy nhất.

```
## TAB PANEL

# Lọc dữ liệu và chỉ lấy các cột cần thiết
df_queens = df.filter(df['neighbourhood_group'] == 'Queens').select('latitude', 'longitude').toPandas()

# Khởi tạo ColumnDataSource từ DataFrame
source_queens = ColumnDataSource(data=df_queens)

# Khởi tạo figure cho Queens
p_queens = figure(title="Tọa độ của Queens", x_axis_label='Longitude', y_axis_label='Latitude')

# Vẽ điểm trên bản đồ cho Queens
p_queens.circle(x='longitude', y='latitude', size=10, color='blue', source=source_queens)

# Khởi tạo figure cho Brooklyn
p_brooklyn = figure(title="Tọa độ của Brooklyn", x_axis_label='Longitude', y_axis_label='Latitude')

# Lọc dữ liệu cho Brooklyn
df_brooklyn = df.filter(df['neighbourhood_group'] == 'Brooklyn').select('latitude', 'longitude').toPandas()

# Khởi tạo ColumnDataSource từ DataFrame cho Brooklyn
source_brooklyn = ColumnDataSource(data=df_brooklyn)

# Vẽ điểm trên bản đồ cho Brooklyn
p_brooklyn.circle(x='longitude', y='latitude', size=10, color='green', source=source_brooklyn)

# Khởi tạo figure cho Bronx
p_bronx = figure(title="Tọa độ của Bronx", x_axis_label='Longitude', y_axis_label='Latitude')

# Lọc dữ liệu cho Bronx
df_bronx = df.filter(df['neighbourhood_group'] == 'Bronx').select('latitude', 'longitude').toPandas()

# Khởi tạo ColumnDataSource từ DataFrame cho Bronx
source_bronx = ColumnDataSource(data=df_bronx)

# Vẽ điểm trên bản đồ cho Bronx
p_bronx.circle(x='longitude', y='latitude', size=10, color='red', source=source_bronx)

# Tạo các tab panel
tab_queens = TabPanel(child=p_queens, title="Queens")
tab_brooklyn = TabPanel(child=p_brooklyn, title="Brooklyn")
tab_bronx = TabPanel(child=p_bronx, title="Bronx")

# Tạo các tabs
tabs = Tabs(tabs=[tab_queens, tab_brooklyn, tab_bronx])

# Hiển thị tab panel
output_notebook()
show(tabs)
```

Trong code trên, chúng ta đang tạo các tab panel để chứa các biểu đồ hoặc bố cục khác. Mỗi tab panel có một tiêu đề tương ứng và chứa một đối tượng con (child), trong trường hợp này là các biểu đồ hoặc bố cục được gán cho các khu vực khác nhau (Queens, Brooklyn, Bronx).

Sau đó, chúng ta tạo các tabs bằng cách sử dụng Tabs object và truyền vào danh sách các tab được tạo trước đó (tab_queens, tab_brooklyn, tab_bronx). Điều này tạo ra một giao diện với các tab có thể được chuyển đổi để hiển thị nội dung của từng khu vực tương ứng.

Hình minh họa :



Hình 7.1 – Biểu đồ 'khu vực của Queens,Brooklyn,Bronx"

Khác với các biểu đồ trước, ta thấy trên góc trái biểu đồ có các nút để chuyển qua các biểu đồ khác.

Slider

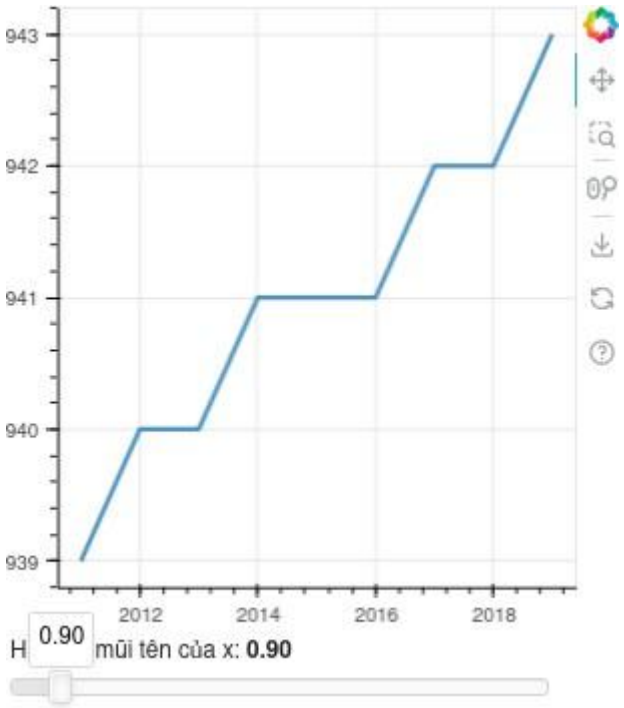
Slider là một thanh trạng thái đồ họa điều khiển giá trị bằng cách di chuyển nó trên một thang đo ngang. Chúng ta có thể thay đổi các giá trị của biểu đồ mà không ảnh hưởng đến định dạng của nó.

```
# Hiển thị kết quả trong notebook
output_notebook()

# Chuyển đổi cột 'last_review' thành năm
airbnb = df.withColumn('year', year(to_date(df['last_review'])))

# Nhóm theo năm và tính tổng giá
grouped_df = df.groupby('year').agg({'price': 'sum'}).orderBy('year')

# Chuyển đổi PySpark DataFrame thành Pandas DataFrame
pandas_df = grouped_df.toPandas()

# Tạo một ColumnDataSource
source = ColumnDataSource(data={"x_values": pandas_df['year'], "y_values": pandas_df['sum(price)']})

# Khởi tạo đối tượng Figure
fig = figure(width=350, height=350)

# Tạo biểu đồ đường
fig.line('x_values', 'y_values', source=source, line_width=2.5, line_alpha=0.8)

# Tạo một callback bằng CustomJS
callback = CustomJS(args=dict(source=source), code="""
    var data = source.data;
    var f = cb_obj.value;
    var x_values = data['x_values'];
    var y_values = data['y_values'];
    for (var i = 0; i < x_values.length; i++) {
        y_values[i] = Math.pow(x_values[i], f);
    }
    source.change.emit();
""")

# Tạo một Slider
slider_widget = Slider(start=0.0, end=10, value=1, step=.1, title="Hiển thị mũi tên của x")
slider_widget.js_on_change('value', callback)

# Tạo layout
slider_widget_layout = column(fig, slider_widget)

# Hiển thị layout
show(slider_widget_layout)
```

ở đoạn code trên, ta sử dụng *Slider()* với các giá trị bắt đầu(start) =0,end = 10, giá trị ban đầu là 1, mỗi bước tăng giảm 1 giá trị và title tên biểu đồ. Mỗi khi mình thao tác với thanh trược thì giá trị callback sẽ

thay đổi .

hình ảnh minh họa :



Hình7.2 – Biểu đồ 'biểu thị giá từng năm"

7. Kết luận

Tổng kết:

- Sử dụng thư viện Matplotlib, Seaborn, Pandas và Bokeh để thực hiện các tác vụ trực quan hóa dữ liệu.
- Tiến hành phân tích và biểu diễn dữ liệu bằng các biểu đồ đường, biểu đồ phân phối, biểu đồ tương quan, và các loại biểu đồ khác.
- Tạo các biểu đồ có tính tương tác để người dùng có thể tương tác và khám phá dữ liệu.
- Cung cấp tiêu đề, nhãn trục và chú thích phù hợp để giúp người xem hiểu dễ

dàng hơn về dữ liệu được biểu diễn.

- Sử dụng các công cụ như Slider, Checkbox, Tabs, và Widgets để tạo ra các tương tác người dùng trong việc thay đổi biểu đồ tại runtime.
- Tạo ra các biểu đồ phù hợp với yêu cầu cụ thể và điều chỉnh các thông số như màu sắc, kích thước, vị trí, và tựa đề để tạo ra các biểu đồ trực quan và dễ hiểu.

Nhận xét:

Ưu điểm :

Bước đầu tiên trong việc sử dụng dữ liệu và thực hiện trực quan hóa là để có cái nhìn tổng quan về dữ liệu, hiểu các xu hướng và mối quan hệ giữa các biến khác nhau.

Nhược điểm :

Tiếp theo là phải áp dụng các phân tích và xử lý dữ liệu cụ thể để trả lời các câu hỏi cụ thể hoặc đưa ra dự đoán chính xác hơn. Hiệu suất của các biểu đồ phụ thuộc vào chất lượng và sẵn có của dữ liệu. Dữ liệu không chính xác hoặc thiếu sót có thể dẫn đến suy luận sai.

Hướng phát triển :

Sẽ cần sử dụng thêm các thư viện về máy học như TensorFlow, PyTorch hoặc Scikit-learn để xây dựng và huấn luyện các mô hình học máy phức tạp hơn.

Hay là sử dụng các công cụ như Plotly và Dash để tăng thêm tính tương tác.

8. Tài liệu tham khảo

Avinash Navlani, Armando Fandango, Ivan Idris. Python Data Analysis, Third Edition, 2021. Packt Publishing.