

# AI & NLP

to Enhance Language Skills



Dataset Airbnb ครอบคลุมสถานที่ต่างๆ คอลเลกชันนี้ให้ข้อมูลเชิงลึก เกี่ยวกับตัวเลือกที่พักประเภทต่างๆ ที่มีให้บริการบนแพลตฟอร์ม Airbnb ด้วยข้อมูลที่ครอบคลุมเกี่ยวกับคุณสมบัติของที่พัก ราคา รีวิว และลักษณะของเจ้าของที่พัก สบนองความอยากรู้อยากเห็นเกี่ยวกับแนวโน้มการเดินทางทั่วโลก

# PREPROCESSING

```
import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('Airbnb_Data.csv')
df.sample(10)
```

id log_price property_type room_type				amenities accommodates bathrooms bed_type cancellation_policy cleaning_fee ...								
25058	2875019	4.007333	Apartment	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet","A...	2	1.0	Real Bed	moderate	False	...	
46314	7507613	4.595120	Bed & Breakfast	Private room	{Internet,"Wireless Internet",Kitchen,"Buzzer/...	3	0.0	Real Bed	flexible	False	...	
63239	13417410	4.094345	Apartment	Shared room	{TV,Internet,"Wireless Internet","Air conditio...	2	1.0	Futon	strict	True	...	

`df.columns`

```
Index(['id', 'log_price', 'property_type', 'room_type', 'amenities',
       'accommodates', 'bathrooms', 'bed_type', 'cancellation_policy',
       'cleaning_fee', 'city', 'description', 'first_review',
       'host_has_profile_pic', 'host_identity_verified', 'host_response_rate',
       'host_since', 'instant_bookable', 'last_review', 'latitude',
       'longitude', 'name', 'neighbourhood', 'number_of_reviews',
       'review_scores_rating', 'thumbnail_url', 'zipcode', 'bedrooms', 'beds'],
      dtype='object')
```

```
df = df.drop(['first_review', 'last_review', 'latitude', 'longitude', 'thumbnail_url', 'zipcode', 'description'],  
df
```

# CLEAN DATA

```
df.isnull().sum()

id
log_price
property_type
room_type
amenities
accommodates
bathrooms
bed_type
cancellation_policy
cleaning_fee
city
host_has_profile_pic
host_identity_verified
host_response_rate
host_since
instant_bookable
name
neighbourhood
number_of_reviews
review_scores_rating
bedrooms
beds
dtype: int64
```

# 1. ลับคอลัมบ์ที่ไม่จำเป็น

## ลับคอลัมบ์ที่ไม่จำเป็นต่อการวิเคราะห์

# ลับคอสัมบ์ที่ไม่จำเป็นต่อการวิเคราะห์

## 2. แทนค่า NULL ในคอลัมน์ NEIGHBOURHOOD ด้วย 'UNKNOWN'

แทนค่า null ด้วย Unknown เพื่อป้องกันไม่ให้เวลาใช้ค่าจากคอลัมบ์ neighbourhood มาวิเคราะห์แล้วเกิด Error

```
df.loc[:, 'neighbourhood'] = df.loc[:, 'neighbourhood'].fillna('Unknown')
```

# 3. เคลียร์ค่า NULL VALUES

ลับແລວທີ່ມີຄ່າໃນບາງຈ່ອງເປັນ null ເພື່ອປ້ອງກັນໄມ້ໃຫ້ເວລາໃຊ້ຄ່າຈາກຄອລັມບັນມາວິເຄຣະໜໍແລ້ວເກີດ Error  
ເນື່ອງຈາກໄມ້ສາມາຮັກຫາຄ່າມາແກນທີ່ໄດ້ໄມ້ເງັນຈະສ່ງຜລຕ່ອກຮິເຄຣະໜໍຂ້ອມູລ ກຳໃຫ້ຂ້ອມູລກັ້ງມັດຈາກ  
73579 ຂ້ອມູລເහັນ 47787 ຂ້ອມູລ

# เปลี่ยนคอลัมม์ HOST SINCE เป็นประเภท TIMESTAMP

เปลี่ยนคอลัมม์ host since จากประเภท Object เป็น timestamp เพื่อให้ง่ายต่อการใช้งานในการวิเคราะห์

```
df["host_since"] = pd.to_datetime(df["host_since"])
```

# เปลี่ยนประเภทจาก OBJECT เป็น CATEGORICAL

เปลี่ยนประเภทจาก Object เป็น Categorical เพื่อให้ง่ายต่อการใช้งานในการวิเคราะห์

```
columns = df.columns
columns

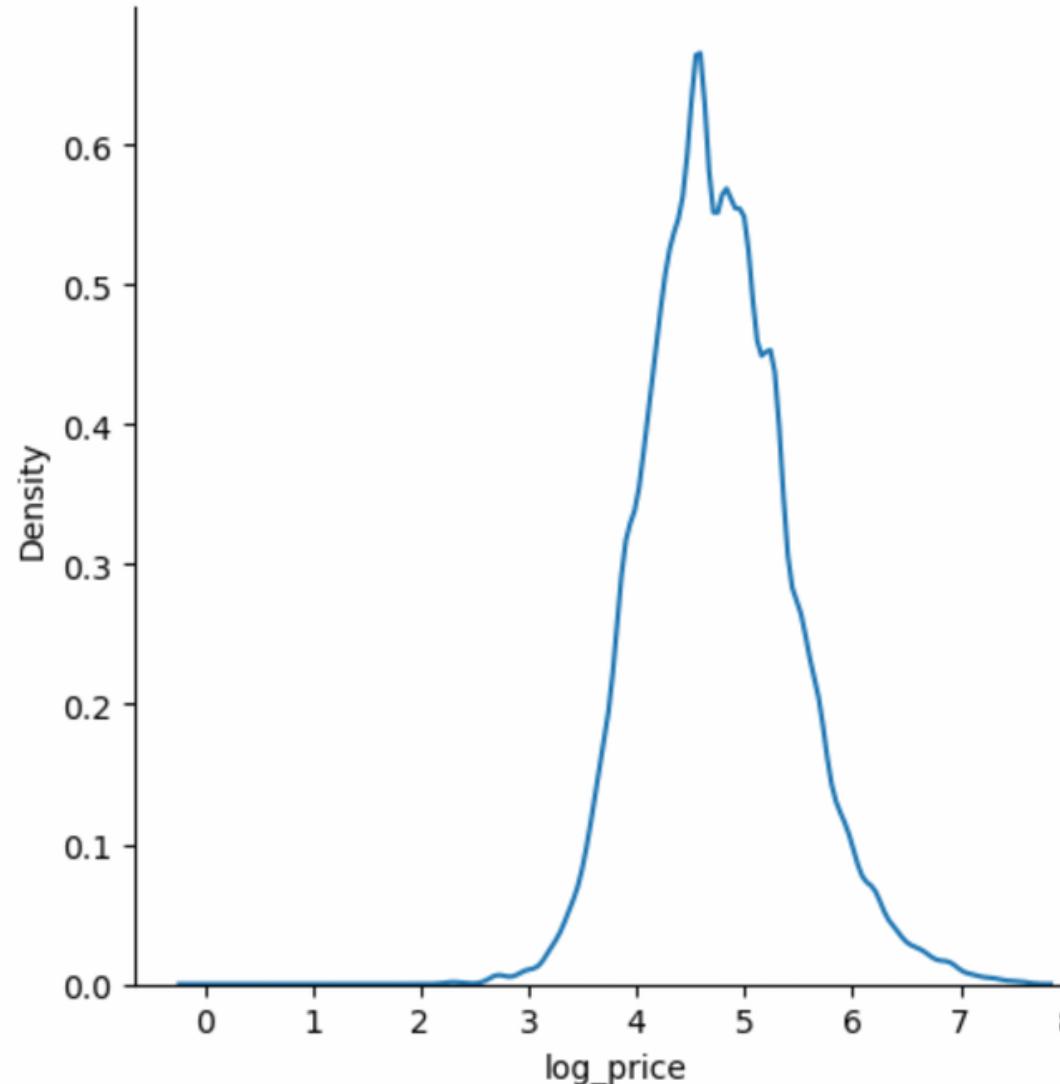
Index(['id', 'log_price', 'property_type', 'room_type', 'amenities',
       'accommodates', 'bathrooms', 'bed_type', 'cancellation_policy',
       'cleaning_fee', 'city', 'host_has_profile_pic',
       'host_identity_verified', 'host_response_rate', 'host_since',
       'instant_bookable', 'name', 'neighbourhood', 'number_of_reviews',
       'review_scores_rating', 'bedrooms', 'beds'],
      dtype='object')

for col in columns:
    if (df[col].dtypes == 'O'):
        df[col] = pd.Categorical(df[col])
```

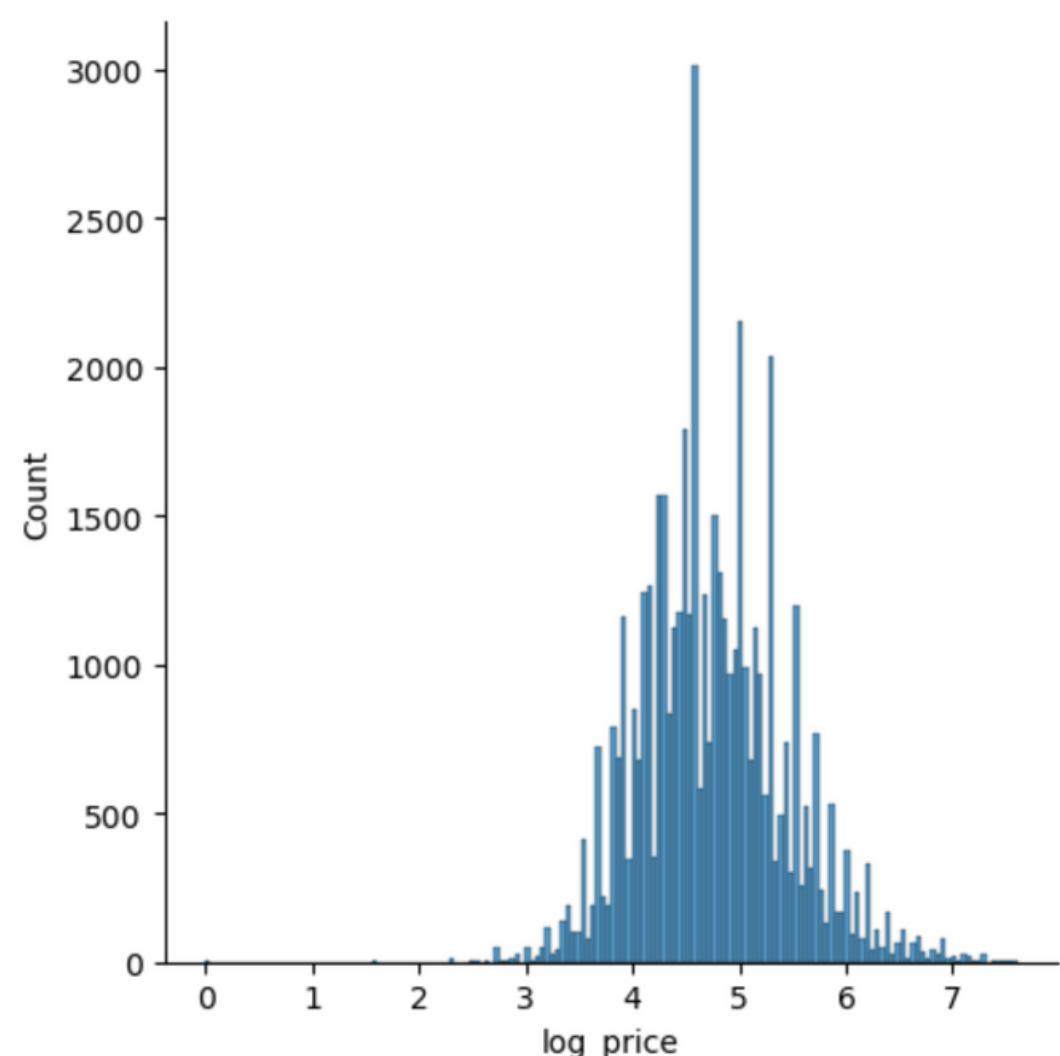
# DESCRIPTIVE STATISTICS

## 1. การกระจายตัวของราคาที่พัสดุ

```
sns.displot(df, x='log_price', kind='kde')  
  
/Users/win/anaconda3/lib/python3.11/site-packages/seaborn/  
ed to tight  
    self._figure.tight_layout(*args, **kwargs)  
  
<seaborn.axisgrid.FacetGrid at 0x15c97f390>
```



```
sns.displot(df, x='log_price', kind='hist')  
  
/Users/win/anaconda3/lib/python3.11/site-packages/seaborn/  
ed to tight  
    self._figure.tight_layout(*args, **kwargs)  
  
<seaborn.axisgrid.FacetGrid at 0x1524ccd50>
```



จากการจะเห็นว่าราคาที่พัสดุส่วนใหญ่มีราคายู่ระหว่าง  $10^3$  ถึง  $10^6$

## 2. ค่าเฉลี่ยของราคาที่พัก

```
df["log_price"].mean()
```

4.751301963982204

## 4. ค่าความแปรปรวนของราคาที่พัก

```
df["log_price"].std()
```

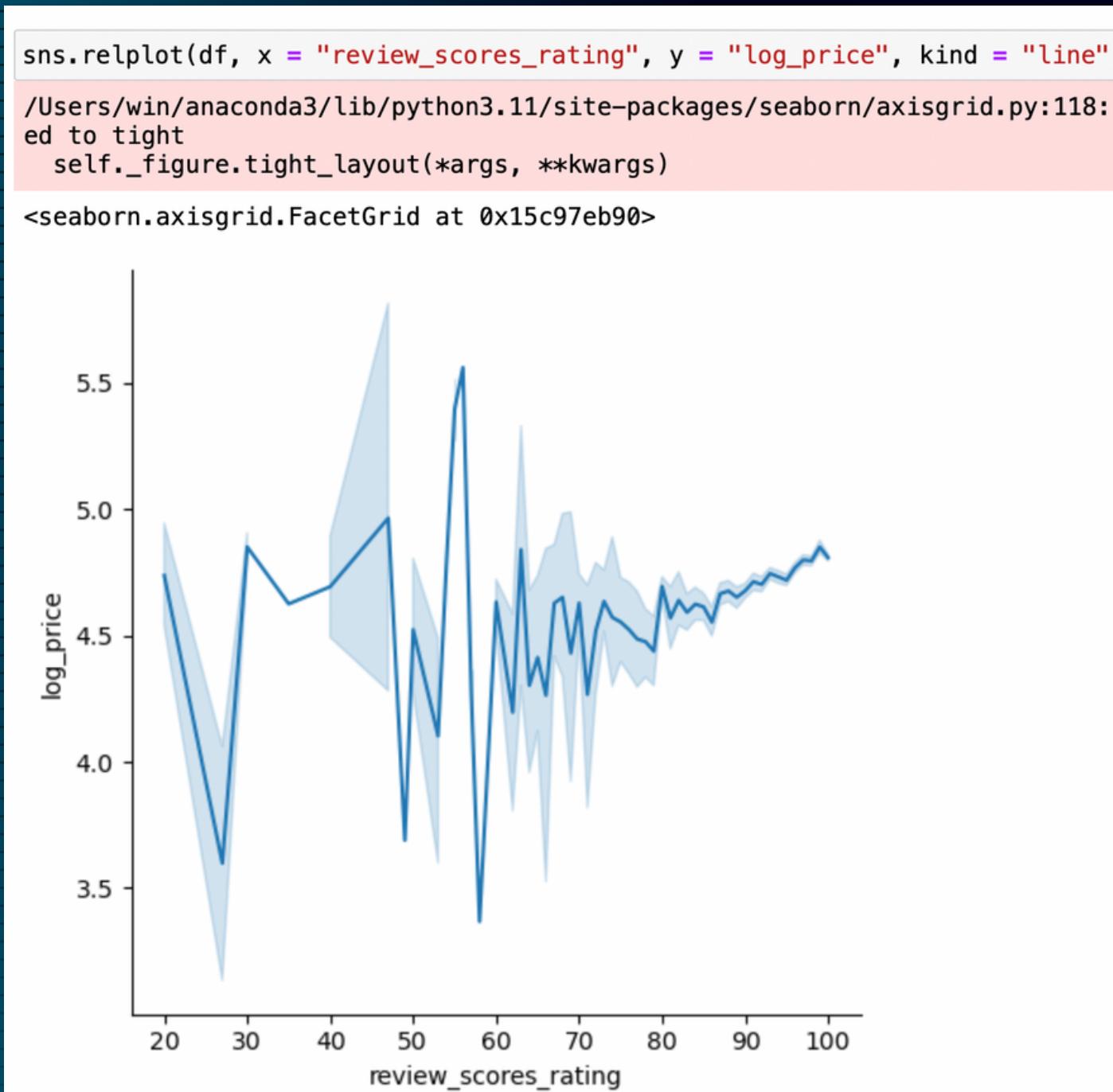
0.675706289385357

## 3. ค่าส่วนเบี่ยงมาตรฐานของราคาที่พัก

```
df["log_price"].var()
```

0.4565789895149278

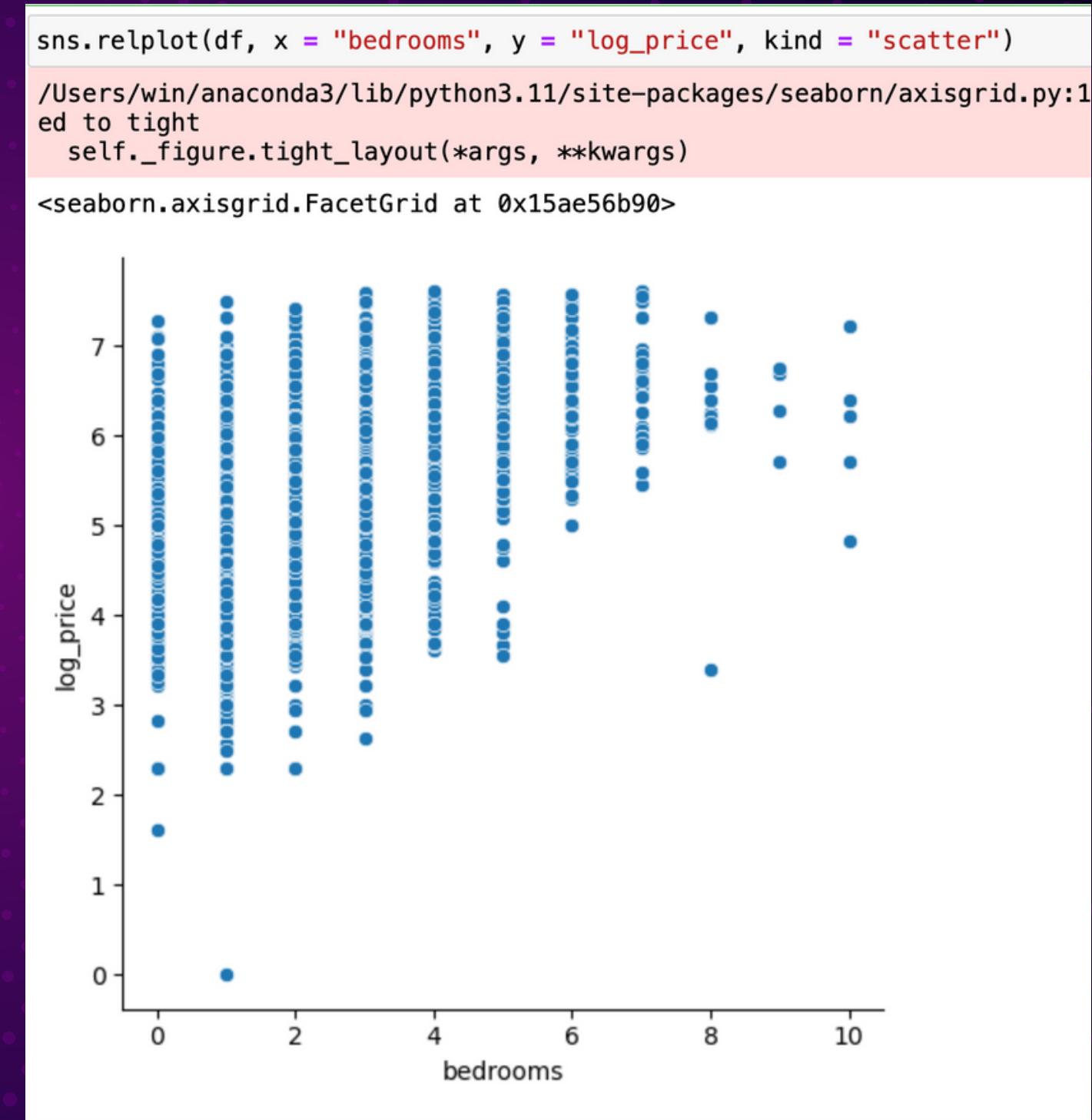
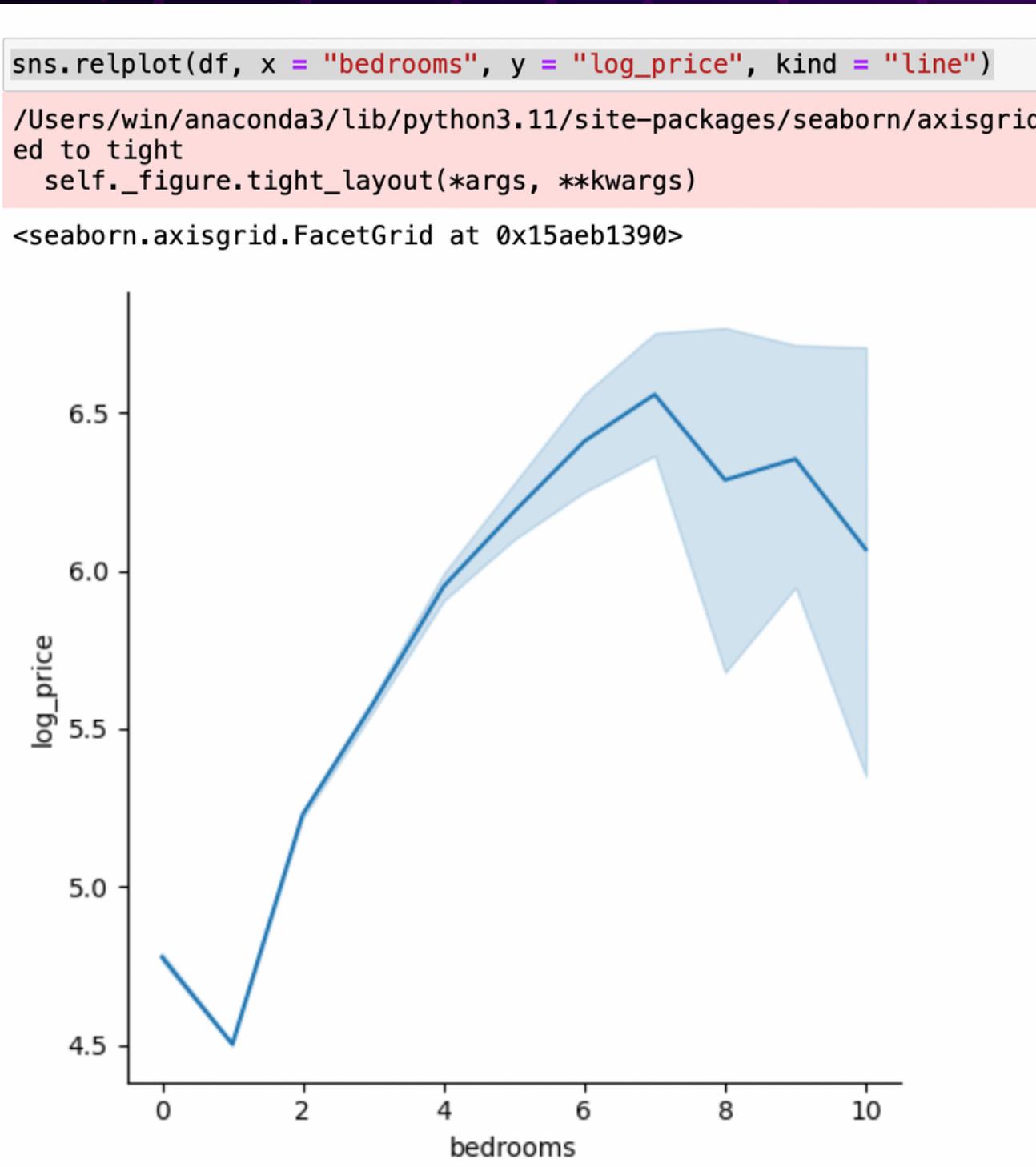
## 5. ความสัมพันธ์ของราคาที่พักกับคะแนนรีวิว



```
df["review_scores_rating"].corr(df.log_price, method = "spearman")  
0.080195463779835  
  
df["log_price"].corr(df["review_scores_rating"], method = "pearson")  
0.07926846262953231
```

กราฟมีแนวโน้มว่าที่พักที่มีคะแนนรีวิวสูงมากมีราคาที่สูงกว่าคะแนนรีวิวและราคาที่พักมีความสัมพันธ์เชิงบวกค่าสัมประสิทธิ์สหสัมพันธ์ระหว่างคะแนนรีวิวและราคาที่พักอยู่ที่ 0.08 หมายความว่า คะแนนรีวิวมีผลต่อราคาที่พักเพียงเล็กน้อย

## ๕. ความสัมพันธ์ระหว่างราคาที่พักกับจำนวนห้องนอน



```
df["log_price"].corr(df["bedrooms"], method = "spearman")
```

0.43567839498647765

```
df["log_price"].corr(df["bedrooms"], method = "pearson")
```

0.5034055360300418

กราฟมีแนวโน้มว่าที่พักที่มีจำนวนห้องนอนมากมักมีราคาที่สูงกว่า  
จำนวนห้องนอนและราคาที่พักมีความสัมพันธ์เชิงบวก  
ค่าสัมประสิทธิ์สหสัมพันธ์ระหว่างจำนวนห้องนอนกับราคามีค่าเฉลี่ยของบ้านคือ 0.50 หมายความว่า  
จำนวนห้องนอนมีผลต่อราคาที่พักที่ปานกลาง

# 7. การทดสอบสมมติฐาน

```
mean_lprice_by_bedrooms = df.groupby('bedrooms').log_price.mean()
mean_lprice_by_bedrooms

bedrooms
0.0    4.777428
1.0    4.502395
2.0    5.228880
3.0    5.578561
4.0    5.949800
5.0    6.186203
6.0    6.409382
7.0    6.557992
8.0    6.287182
9.0    6.353662
10.0   6.067557
Name: log_price, dtype: float64
```

H<sub>0</sub>: ห้องนอน 6 ห้อง ราคามากเท่ากับ ห้องนอน 7 ห้อง ที่นัยสำคัญ 0.1

H<sub>1</sub>: ห้องนอน 7 ห้อง ราคามากกว่า ห้องนอน 6 ห้อง จริง

```
alpha = 0.1
```

```
(diff_log_price_bedrooms >= real_diff).sum() / 1000 #p-value
0.141
```

กลุ่มตัวอย่าง

```
real_diff = mean_lprice_by_bedrooms[7] - mean_lprice_by_bedrooms[6]
real_diff
0.14861073675376968
```

```
num_by_bedrooms = df.groupby('bedrooms').size()
num_by_bedrooms
```

```
bedrooms
0.0      4409
1.0     31457
2.0      7766
3.0      2922
4.0       878
5.0       240
6.0        63
7.0       31
8.0       12
9.0        4
10.0      5
dtype: int64
```

```
num_grp_A = num_by_bedrooms[6]
num_grp_B = num_by_bedrooms[7]
```

```
df_67 = df.query('bedrooms == 6 | bedrooms == 7')
```

```
ls = []
for i in range(1000):
    grp_A = df_67.sample(num_grp_A)
    grp_B = df_67.loc[ ~df_67.index.isin( grp_A.index ), : ]
    mean_log_price_diff = grp_B.log_price.mean() - grp_A.log_price.mean()
    ls.append(mean_log_price_diff)
```

```
diff_log_price_bedrooms = pd.Series(ls)
diff_log_price_bedrooms
```

p-value สูงกว่า alpha ดังนั้นเราสามารถตัดสินใจ H<sub>1</sub> ได้ เราต้องยอมรับ H<sub>0</sub> ว่าเหตุการณ์ที่เกิดขึ้นเป็นบังเอิญ (ห้องนอน 7 ห้อง ราคามากกว่า ห้องนอน 6 ห้อง บังเอิญ)

## 8. ความสัมพันธ์เชิงเส้นและทิศทาง

หาความสัมพันธ์ของ columns ที่มีแนวโน้มของ spearman หรือ pearson มากกว่า 0.5

```
def pearson_spearman(df, method, ls):
    for i in range(len(df.columns) - 1):
        for j in range(i + 1, len(df.columns)):
            df_i, df_j = df.iloc[:,i], df.iloc[:,j]
            if (df_i.dtype in ['int64', 'float64'] and df_j.dtype in ['int64', 'float64']):
                temp = df_i.corr(df_j, method = method)
                if abs(temp) >= 0.5 and df_i.name + " " + df_j.name not in ls:
                    ls.append(df_i.name + " " + df_j.name)
                    print(df_i.name, df_j.name, df_i.corr(df_j, method = method))

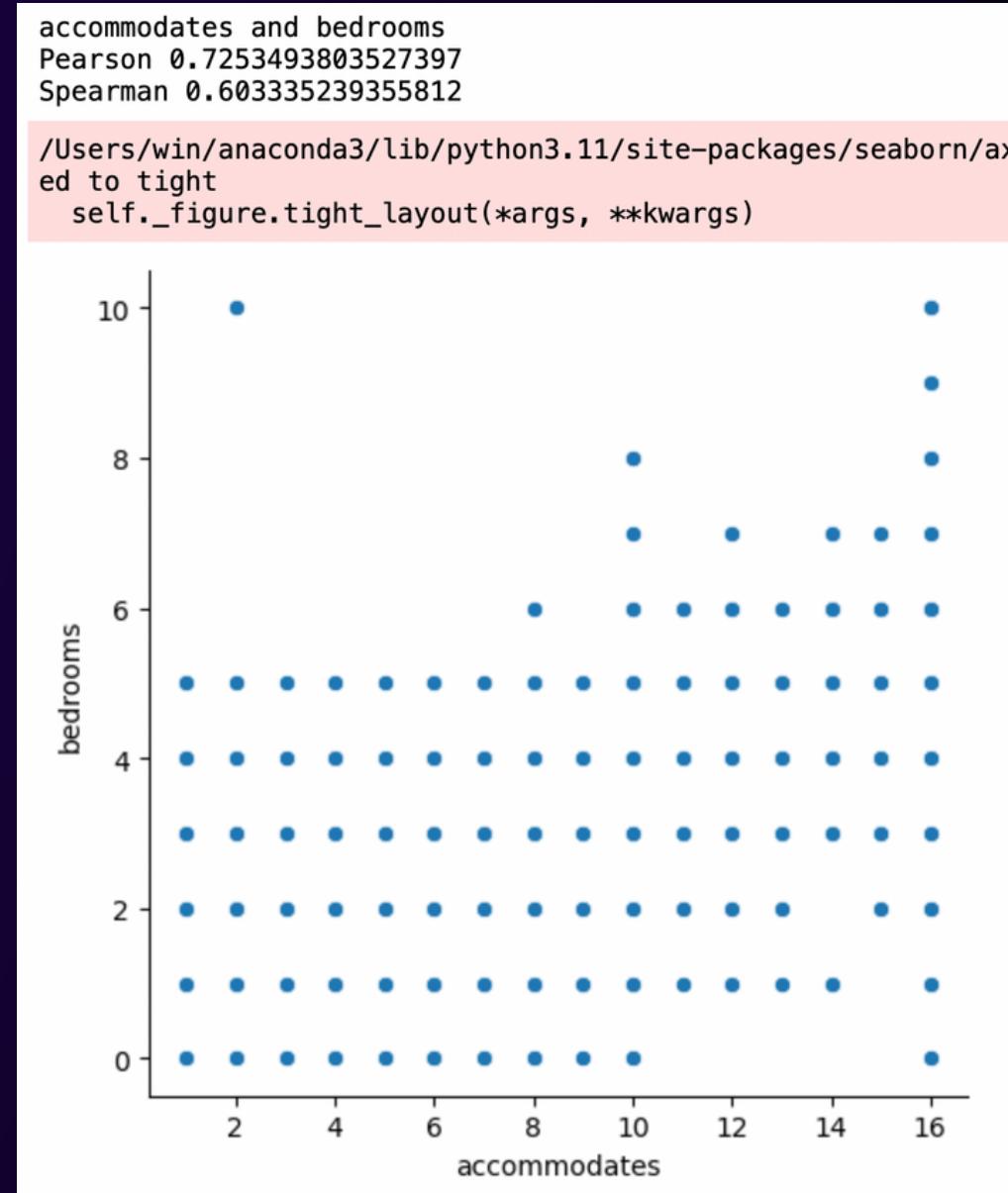
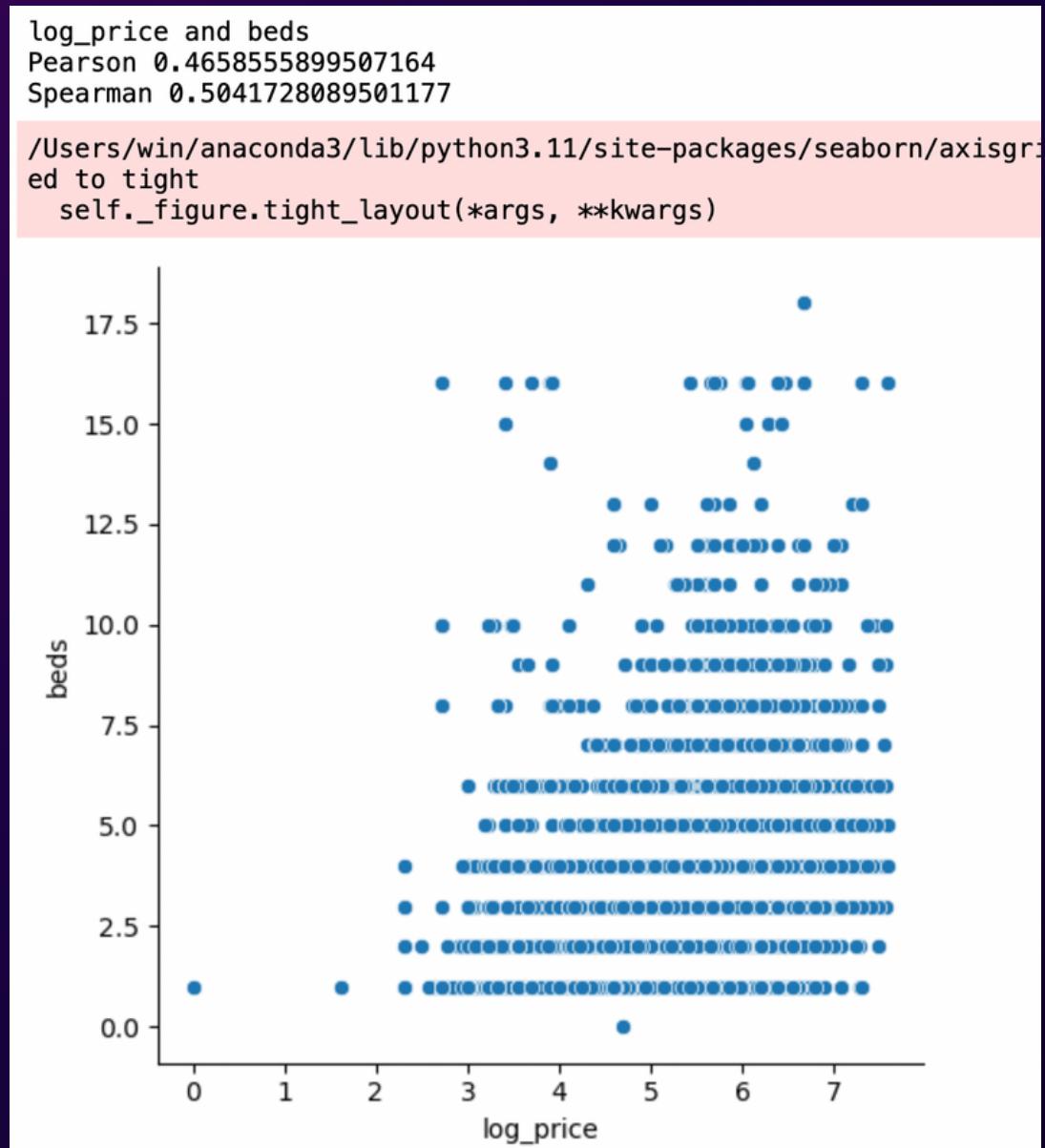
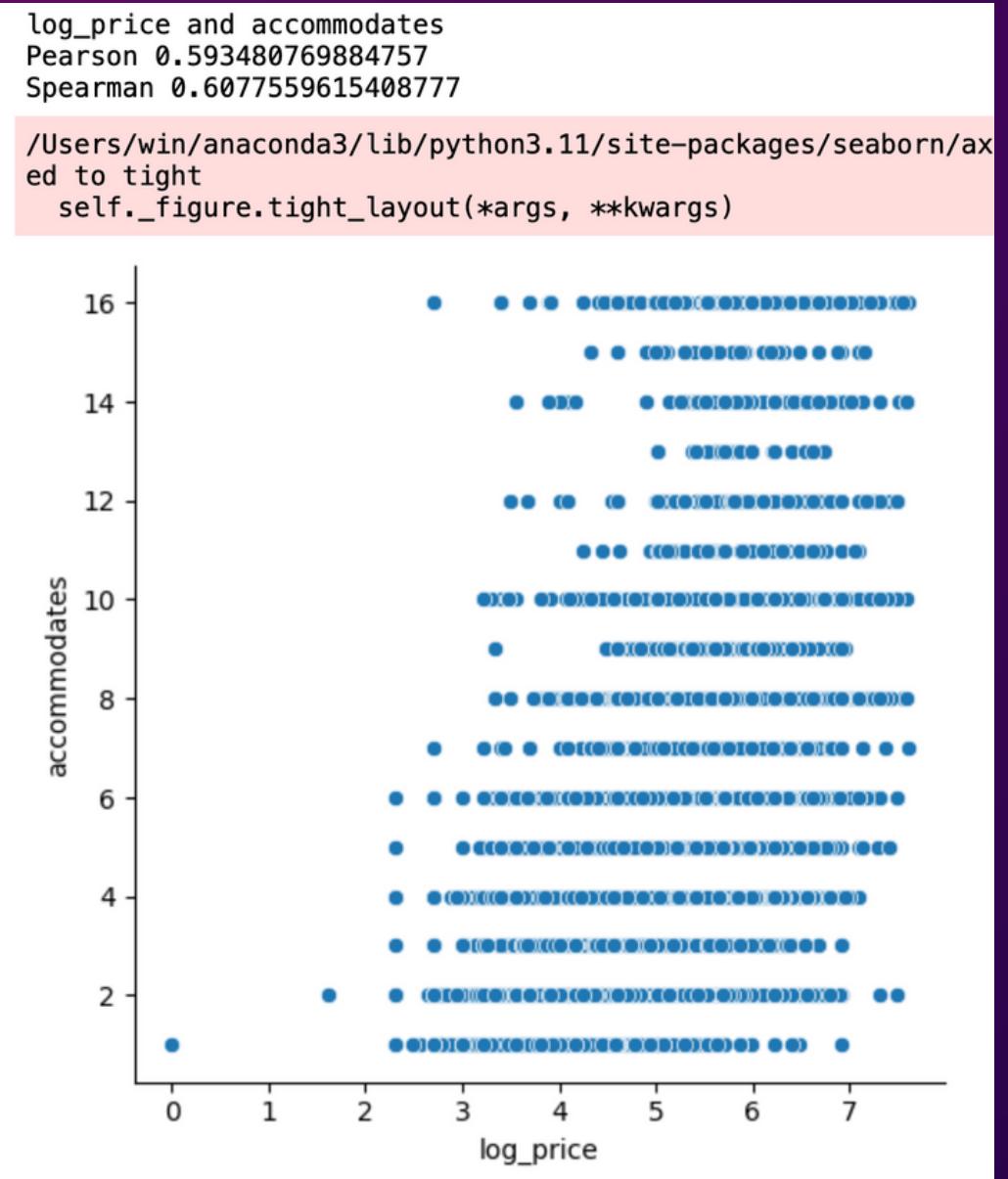
print("---Spearman---")
ls = []
pearson_spearman(df, "spearman", ls)

---Spearman---
log_price accommodates 0.6077559615408777
log_price beds 0.5041728089501177
accommodates bedrooms 0.603335239355812
accommodates beds 0.8084685618247734
bedrooms beds 0.642156690754686

print("---Pearson---")
pearson_spearman(df, "pearson", ls)

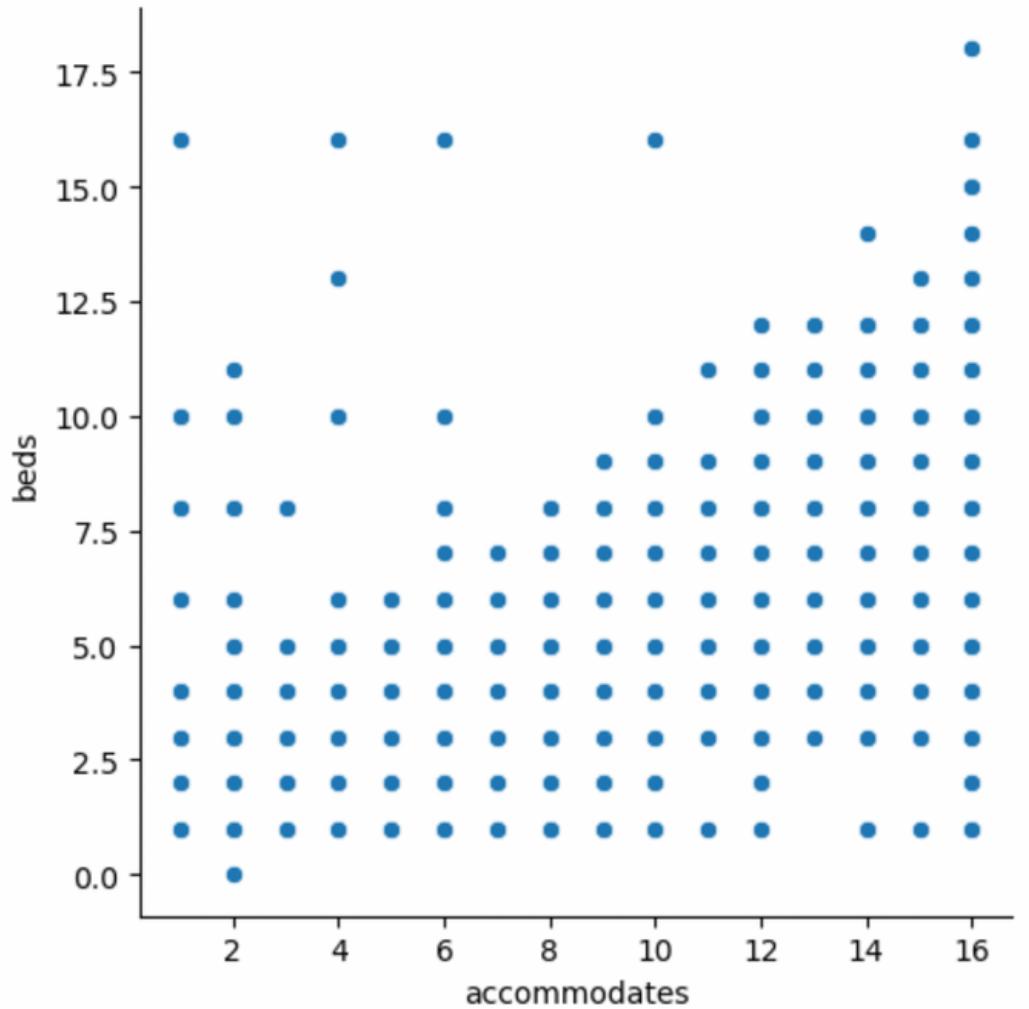
---Pearson---
log_price bedrooms 0.5034055360300418
bathrooms bedrooms 0.5756059415880238
bathrooms beds 0.5111975770189229
```

```
def plot_pearson_seaborn(string):
    i, j = string.split()
    df_i, df_j = df.loc[:,i], df.loc[:,j]
    print(i, 'and', j)
    print('Pearson' ,df_i.corr(df_j, method = "pearson"))
    print('Spearman' ,df_i.corr(df_j, method = "spearman"))
    sns.relplot(df, x = i, y = j, kind = "scatter")
```



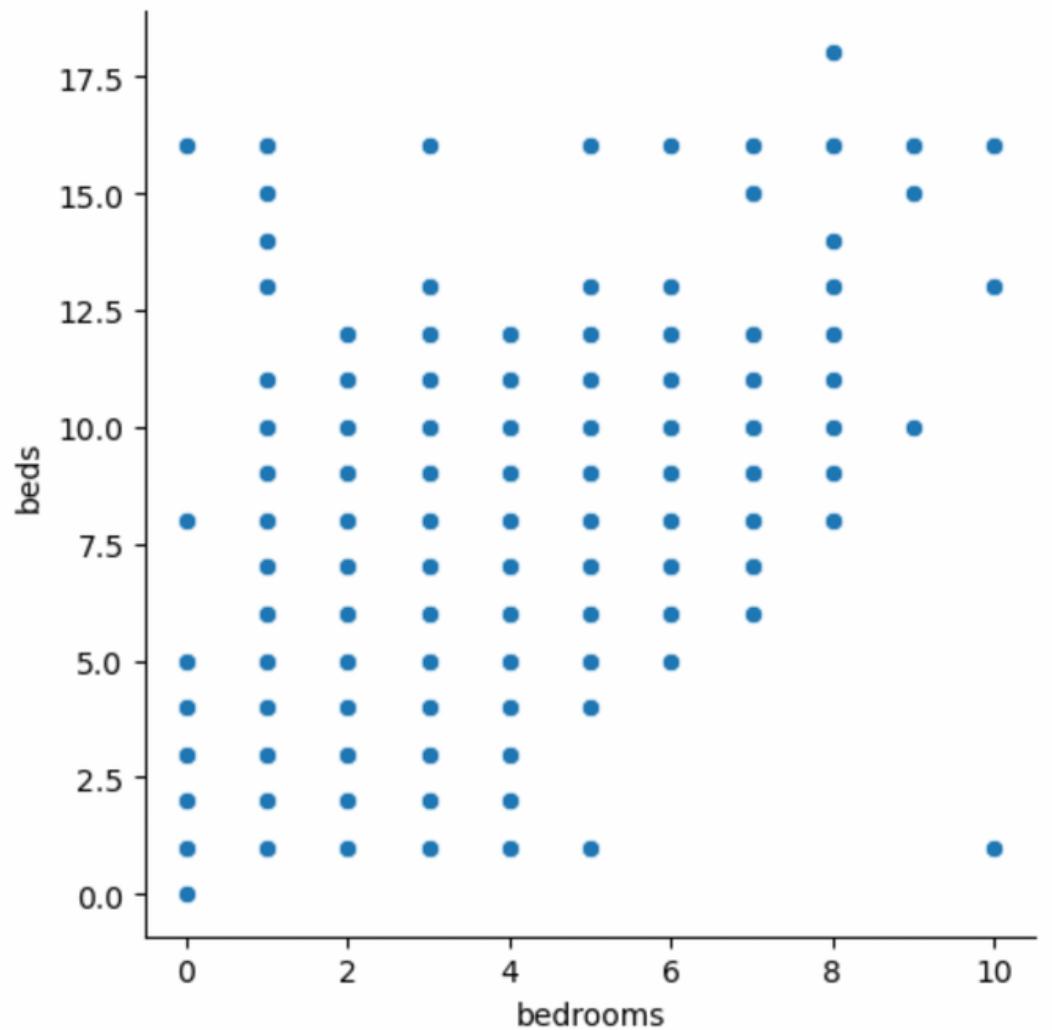
accommodates and beds  
Pearson 0.8251033910955954  
Spearman 0.8084685618247734

/Users/win/anaconda3/lib/python3.11/site-packages/seaborn/\_decorators.py:40: UserWarning: Matplotlib is being used outside the main thread.  
self.\_figure.tight\_layout(\*args, \*\*kwargs)



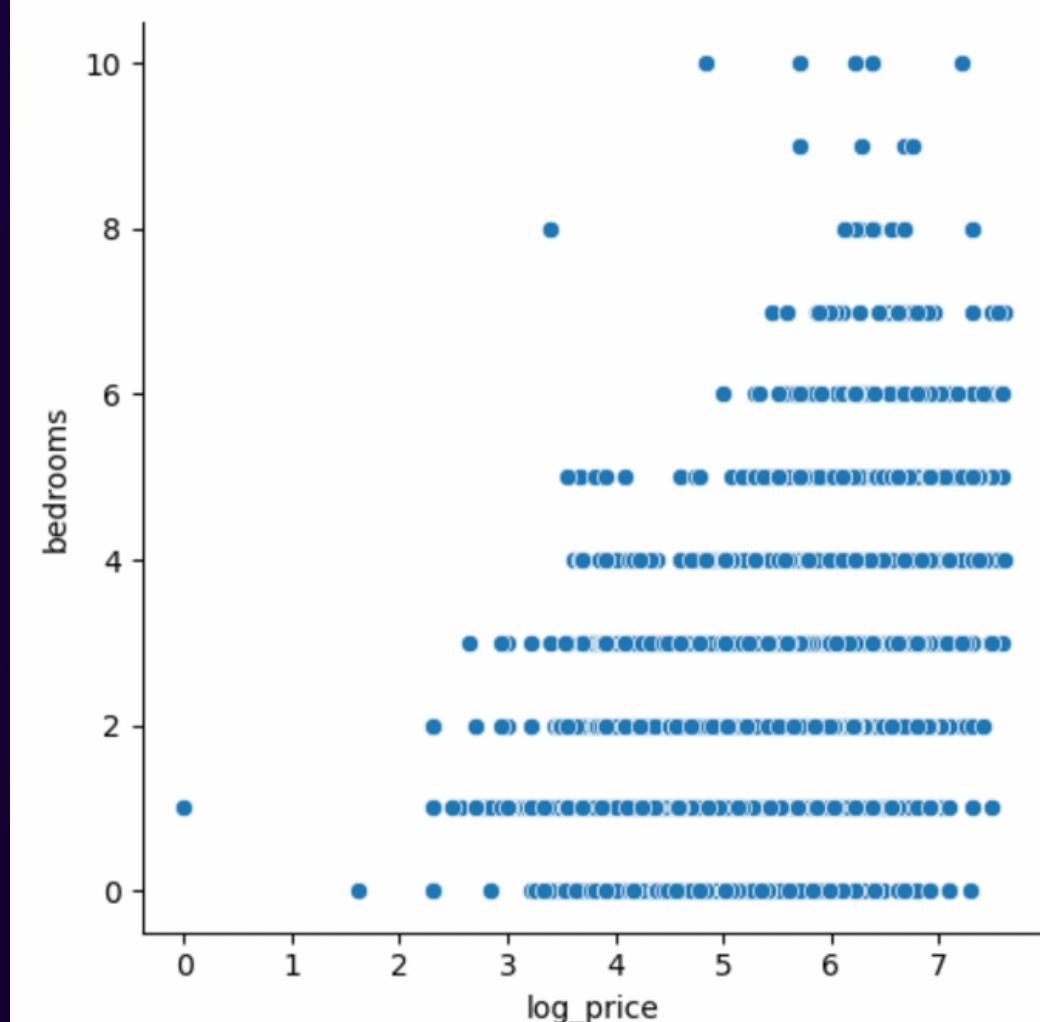
bedrooms and beds  
Pearson 0.7045252069974537  
Spearman 0.642156690754686

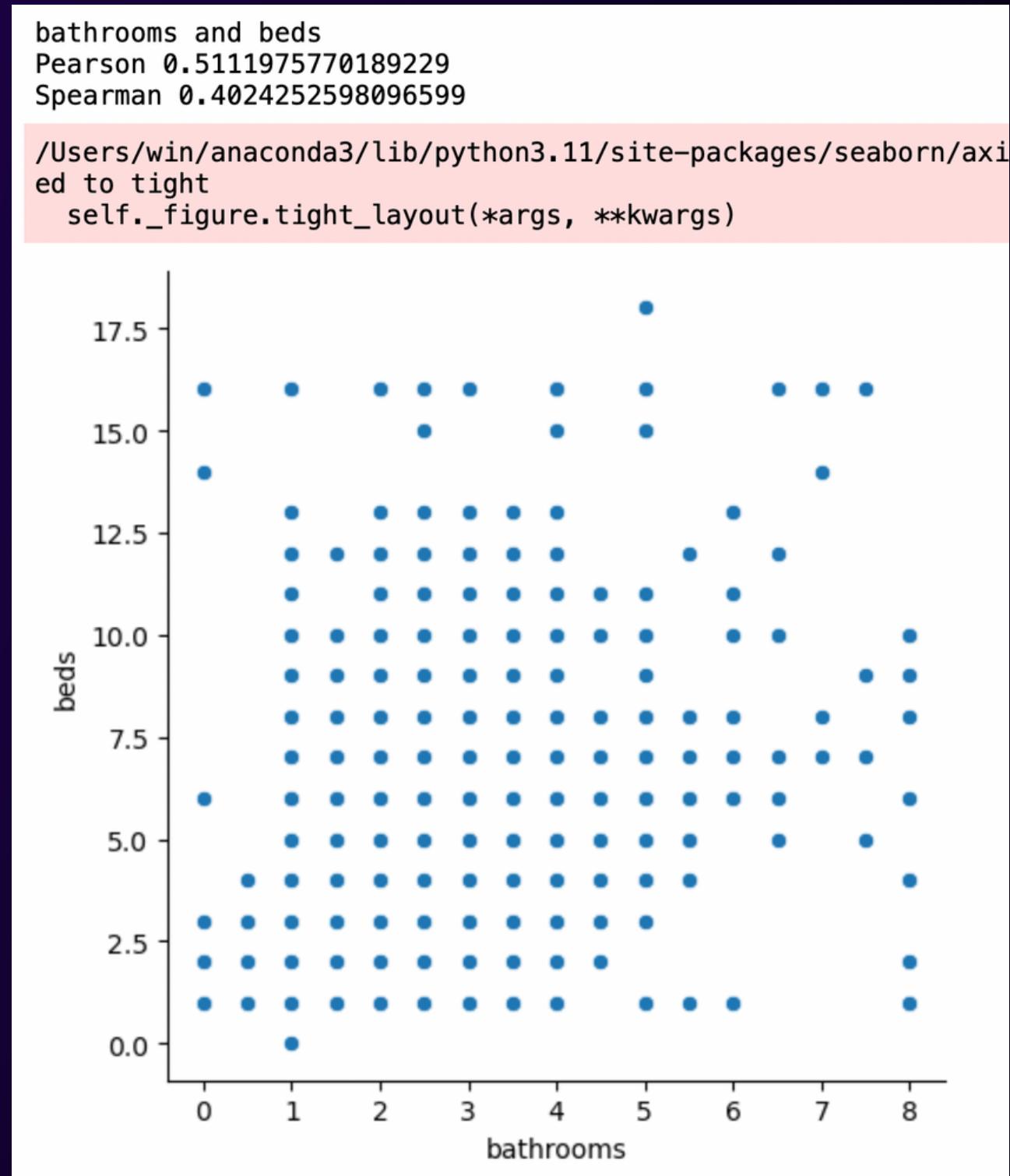
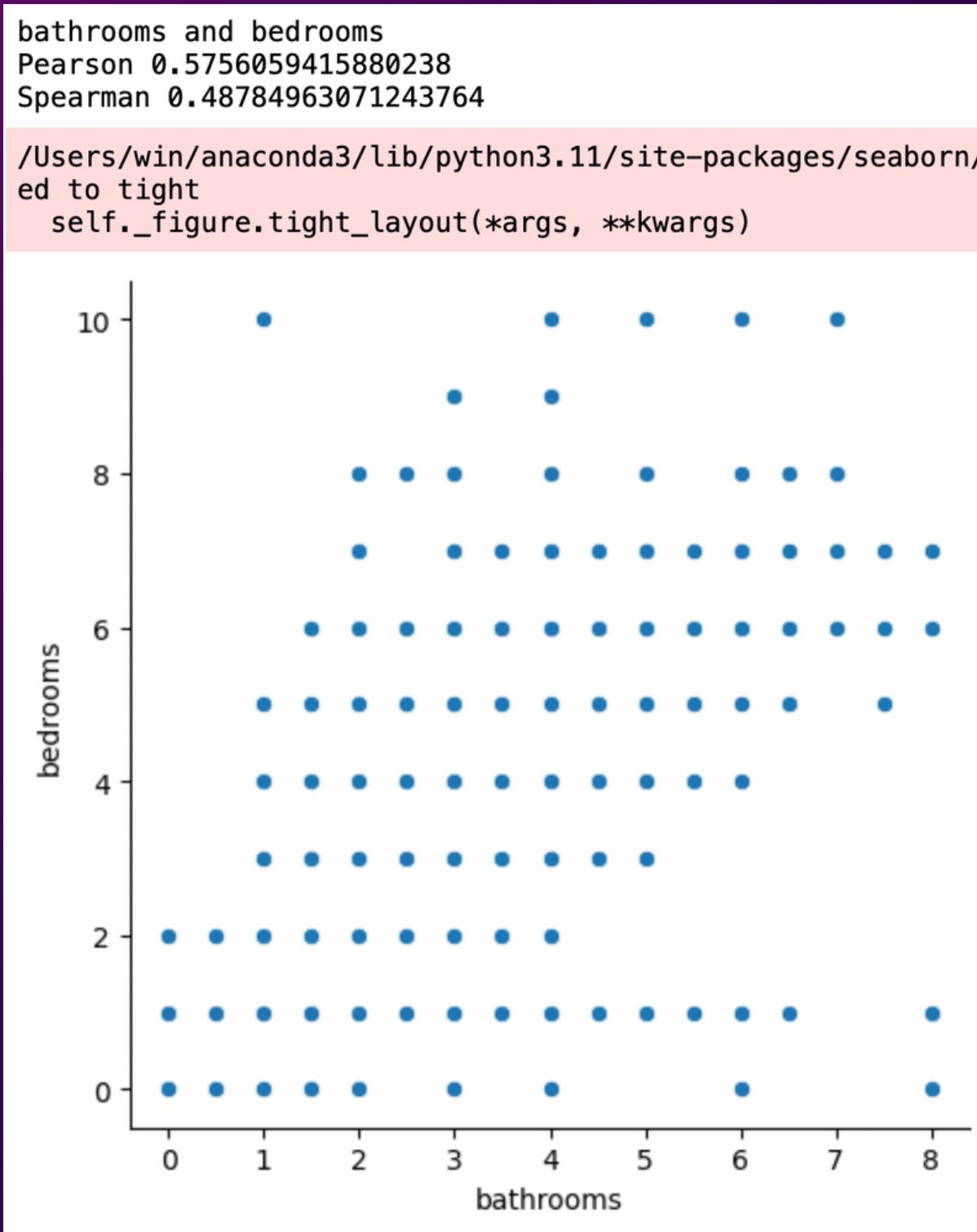
/Users/win/anaconda3/lib/python3.11/site-packages/seaborn/\_decorators.py:40: UserWarning: Matplotlib is being used outside the main thread.  
self.\_figure.tight\_layout(\*args, \*\*kwargs)



log\_price and bedrooms  
Pearson 0.5034055360300418  
Spearman 0.43567839498647765

/Users/win/anaconda3/lib/python3.11/site-packages/seaborn/\_decorators.py:40: UserWarning: Matplotlib is being used outside the main thread.  
self.\_figure.tight\_layout(\*args, \*\*kwargs)





เมื่อวิเคราะห์จากหลายๆ กราฟ และค่าของ pearson, spearman ดังกล่าวพบว่า columns ส่วนใหญ่ที่มีแนวโน้มความชันเป็นบวกจะมีความสัมพันธ์ในทิศทางเดียวกัน

# 9. วิเคราะห์ข้อมูลตามกลุ่มที่สนใจ

## 9.1 วิเคราะห์ค่าเฉลี่ยราคาของที่พักแต่ละประเภทที่พัก

```
property_log_price = df.loc[:, ["property_type", "log_price"]].groupby(["property_type"]).mean()  
property_log_price
```

mean_log_price	
property_type	
Apartment	4.751866
Bed & Breakfast	4.511107
Boat	5.107821
Boutique hotel	4.780398
Bungalow	4.776984
Cabin	4.620320
Camper/RV	4.424473
Castle	5.368424
Cave	4.909373
Chalet	4.848429

จากการวิเคราะห์ประเภทของที่พักไม่ค่อยมีผลต่อจำนวนที่พักสูงสุด  
ราคาประเภทที่พักที่ราคาแพงที่สุดคือ Timeshare

```
max_plp = property_log_price["mean_log_price"].max()  
property_log_price[property_log_price["mean_log_price"] == max_plp]
```

mean_log_price	
property_type	
Timeshare	5.57974

## 9.2 วิเคราะห์ค่าเฉลี่ยจำนวนคนที่เข้าพักได้สูงสุดแต่ละประเภท

```
prop_acc = df.loc[:, ["property_type", "accommodates"]].groupby(["property_type"]).mean()  
prop_acc
```

mean_accommodates	
property_type	
Apartment	3.114431
Bed & Breakfast	2.539157
Boat	3.177778
Boutique hotel	2.945946
Bungalow	3.093190
Cabin	2.406780
Camper/RV	2.923077
Castle	4.846154
Cave	1.500000
Chalet	3.400000

จากการวิเคราะห์ประเภทของที่พักมีผลต่อจำนวนคนที่เข้าพักสูงสุด  
ประเภทที่พักที่เข้าพักได้มากที่สุดคือ Villa

```
max_pa = prop_acc["mean_accommodates"].max()  
prop_acc[prop_acc["mean_accommodates"] == max_pa]
```

mean_accommodates	
property_type	
Villa	5.32

## 9.3 วิเคราะห์ค่าเฉลี่ยราคาแต่ละจำนวนคนที่เข้าพักได้สูงสุด

```
acc_loc = df.loc[:, ["log_price", "accommodates"]].groupby(["accommodates"]).mean()
acc_loc
```

mean_log_price
accommodates
1 4.096264
2 4.501416
3 4.734784
4 4.980269
5 5.163231
6 5.352831
7 5.453122
8 5.637284
9 5.626796
10 5.791251

จากการวิเคราะห์จำนวนคนที่เข้าได้สูงสุดพักมีผลต่อราคา  
จำนวนคนเข้าพักที่ราคาพักที่แพงที่สุด คือ 12 คน

```
max_la = acc_loc["mean_log_price"].max()
acc_loc[acc_loc["mean_log_price"] == max_la]
```

mean_log_price
accommodates
12 6.025528