CA1

February 10, 2023

1 Compulsory Assignment 1 - Pandas and visualizations

1.0.1 Imports

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

1.1 Loading and exploring the dataset

1. Load the dataset named `airbnb.csv` and store it in a dataframe called `raw_df`. Use the column named `id` as the index column for the dataframe

```
[ ]: # Insert your code below
     # ======================

     raw_df = pd.read_csv('assets/airbnb.csv', index_col='id', )
```

2. Print the first `five` rows of the dataframe

```
[ ]: # Insert your code below
     # ======================

     print(raw_df.head(5))
```

```
                                                    name  host_id  host_name  \
id
183319                      Panoramic Ocean View Venice Beach   867995  Barbara X
109     Amazing bright elegant condo park front *UPGRA…      521      Paolo
51307   Spanish Bungalow Guest House LA CA. 30 plus ni…   235568      David
184314                        Boho Chic Flat..Steps to Beach!   884031     Ashley
51498   Guest House With Its Own Entrance/Exit and Hot…   236758        Bay

          neighbourhood  latitude  longitude        room_type  price  \
id
183319           Venice  33.99211 -118.47600  Entire home/apt    152
109         Culver City  33.98301 -118.38607  Entire home/apt    115
```

```
51307   Atwater Village  34.12206 -118.26783  Entire home/apt     75
184314           Venice  33.97487 -118.46312  Entire home/apt    125
51498         Mar Vista  34.00389 -118.44126  Entire home/apt    189

        minimum_nights  number_of_reviews  calculated_host_listings_count  \
id
183319              30                  3                               2
109                 30                  2                               1
51307               30                138                               2
184314              30                 30                               1
51498                3                378                               1

        availability_365  number_of_reviews_ltm state         city
id
183319                 0                      0    CA  Los Angeles
109                  139                      0    CA  Los Angeles
51307                224                      0    CA  Los Angeles
184314                 0                      0    CA  Los Angeles
51498                348                     41    CA  Los Angeles
```

3. How many unique values exist in each of the columns state and city?

[ ]:
```
# Insert your code below
# ======================

state = raw_df.state
city = raw_df.city

unique_state = state.nunique()
unique_city = city.nunique()

print(f'Number of unique states: {unique_state} \n')
print(f'Number of unique cities: {unique_city} \n')
```

```
Number of unique states: 19

Number of unique cities: 31
```

4. Identify missing (NaN) values in each of the columns in the dataset

[ ]:
```
# Insert your code below
# ======================
print(f'Column       Number of missing values ')
for c in raw_df.columns:
    n_NaN = raw_df[c].isnull().sum()
    print(f'{c:<32} {n_NaN}')
```

```
Column       Number of missing values
name                             19
```

```
host_id                           0
host_name                         1144
neighbourhood                     712
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
calculated_host_listings_count    0
availability_365                  0
number_of_reviews_ltm             0
state                             0
city                              0
```

5. Create a copy of `raw_df` named `df`. Remove any rows containing NaN values in the new dataframe. What is the shape of `df` before and after removing the NaN values?

[ ]:
```
# Insert your code below
# ======================

df = raw_df.copy()

df_shape_with_nan = df.shape

df.dropna(inplace=True)

print('The shape before removing NaN values: ', df_shape_with_nan)
print('The shape after removing NaN values: ', df.shape)
```

```
The shape before removing NaN values:  (325858, 15)
The shape after removing NaN values:  (323983, 15)
```

6. Which `room_type`, `state` and `city` is the most popular (by number of instances)? Print the name and count of each

Hint: The output should look something like this:

```
Column: [col], Most popular: [name], Count: [count]
Column: [col], Most popular: [name], Count: [count]
Column: [col], Most popular: [name], Count: [count]
```

[ ]:
```
# Insert your code below
# ======================

room_type = df.room_type
city = df.city
state = df.state

most_frequent_city = city.value_counts().index[0]
```

```
max_count_city = max(city.value_counts())

most_frequent_state = state.value_counts().index[0]
max_count_state = max(state.value_counts())

most_frequent_room_type = room_type.value_counts().index[0]
max_count_room_type = max(room_type.value_counts())

print(f'Column: {city.name:<11} Most popular: {most_frequent_city:<17} Count:␣
 ↪{max_count_city}')
print(f'Column: {state.name:<11} Most popular: {most_frequent_state:<17} Count:␣
 ↪{max_count_state}')
print(f'Column: {room_type.name:<11} Most popular: {most_frequent_room_type:
 ↪<17} Count: {max_count_room_type}')
```

```
Column: city        Most popular: Los Angeles       Count: 91600
Column: state       Most popular: CA                Count: 127206
Column: room_type   Most popular: Entire home/apt   Count: 241433
```

7. What is the average and median price for a listing?

[ ]:
```
# Insert your code below
# ======================

price = df.price

print('The median of the price is: ', price.median())
print('The mean of the price is: ', price.mean())
```

```
The median of the price is:  159.0
The mean of the price is:  285.125163974653
```

8. What is the average price for the states `CA`, `FL` and `NY`?

Hint: The output should look something like this:

```
State: [col], Average price: [price]
State: [col], Average price: [price]
State: [col], Average price: [price]
```

[ ]:
```
# Insert your code below
# ======================

CA = df[df.state == 'CA']
mean_price_CA = CA.price.mean()

FL = df[df.state == 'FL']
mean_price_FL = FL.price.mean()

NY = df[df.state == 'NY']
```

```
mean_price_NY = NY.price.mean()

print(f'State: CA    Average price: {mean_price_CA}')
print(f'State: FL    Average price: {mean_price_FL}')
print(f'State: NY    Average price: {mean_price_NY}')
```

```
State: CA    Average price: 288.39531940317283
State: FL    Average price: 241.98664420647336
State: NY    Average price: 197.21922246220302
```

9. Create a new dataframe called `df_beach` containing all listings with "beach" in the `name`. Print out the shape of `beach_df`

The filtering should not be case sensitive, meaning that names containing `beach`, `Beach`, `BeAcH` etc. all should be included

[ ]:
```
# Insert your code below
# ======================

df_beach = df[df['name'].str.contains('beach', case=False)]

print(df_beach.shape)
```

```
(31436, 15)
```

1.2 Visualizing the dataset

10. Create plot with 2 vertical axes and one horizontal axes. The plot should display a barchart containing the count of the `10 most` popular states and cities, each in its own subplot. The bars should be sorted in descending order.

Use `df` in all tasks in this section

Hint: It is recommended to use the `Barplot` function built into Seaborn for barcharts.

The output should look something like this:

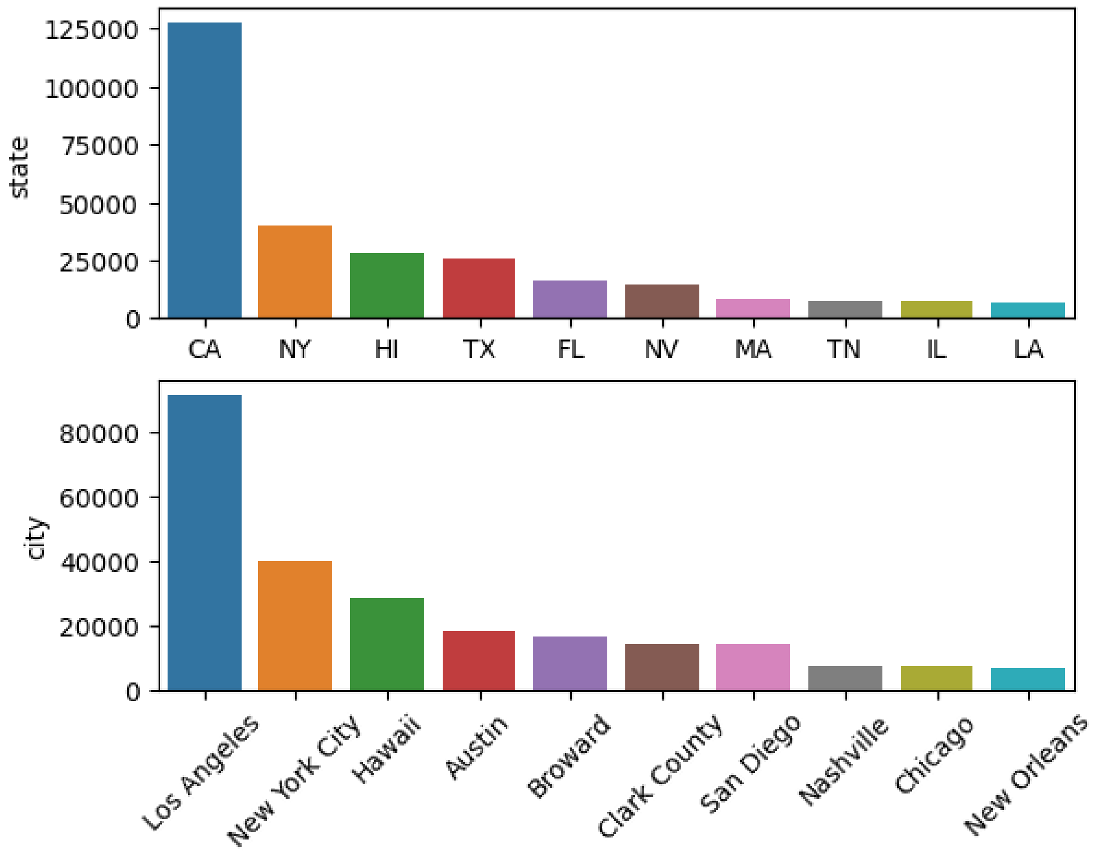
PS: Disregard the color scheme of the example image.

[ ]:
```
# Insert your code below
# ======================

states = df.state.value_counts()[:10]
cities = df.city.value_counts()[:10]

fig, axs = plt.subplots(2)

sns.barplot(ax=axs[0], x=states.index, y=states)
sns.barplot(ax=axs[1], x=cities.index, y=cities)
axs[1].tick_params(axis='x', rotation=45)
```



11. Create a scatterplot with the longitude and latitude of the listings in `df`. Longitude should be on the x-axis and latitude on the y-axis.

The output should look something like this:

PS: Disregard the color scheme of the example image.

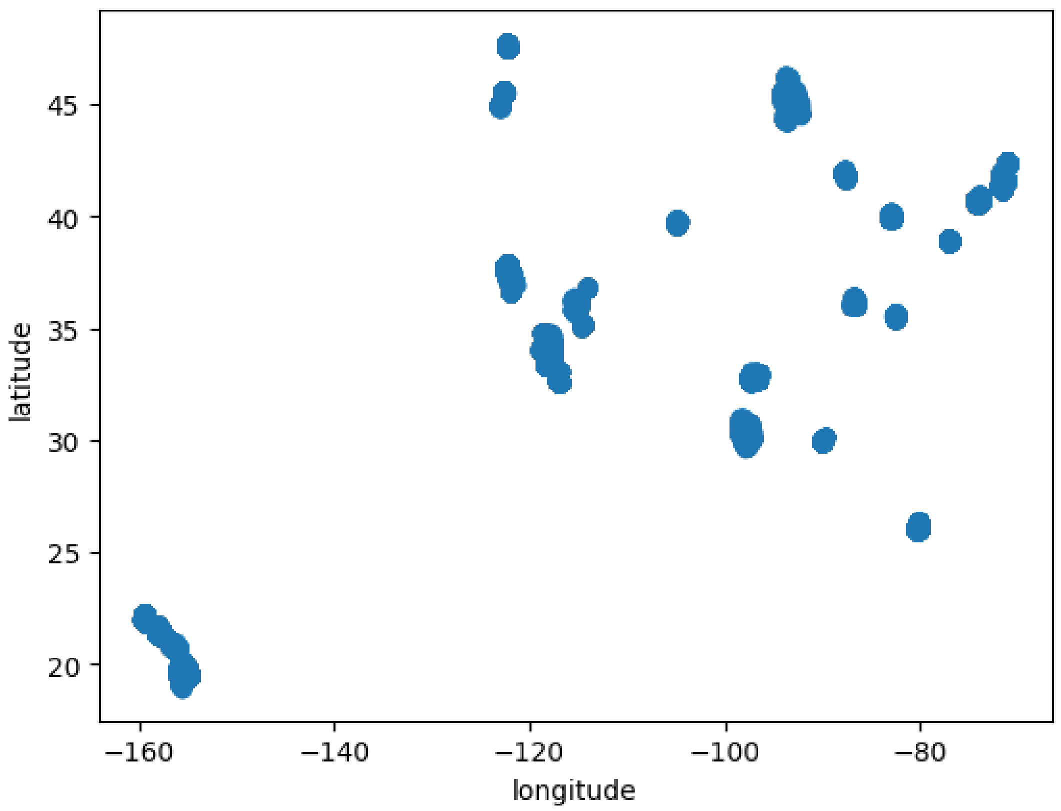
[ ]:
```
# Insert your code below
# ======================

longitude = df.longitude
latitude = df.latitude

plt.scatter(longitude, latitude)

plt.xlabel('longitude')
plt.ylabel('latitude')
```

[ ]:
```
Text(0, 0.5, 'latitude')
```

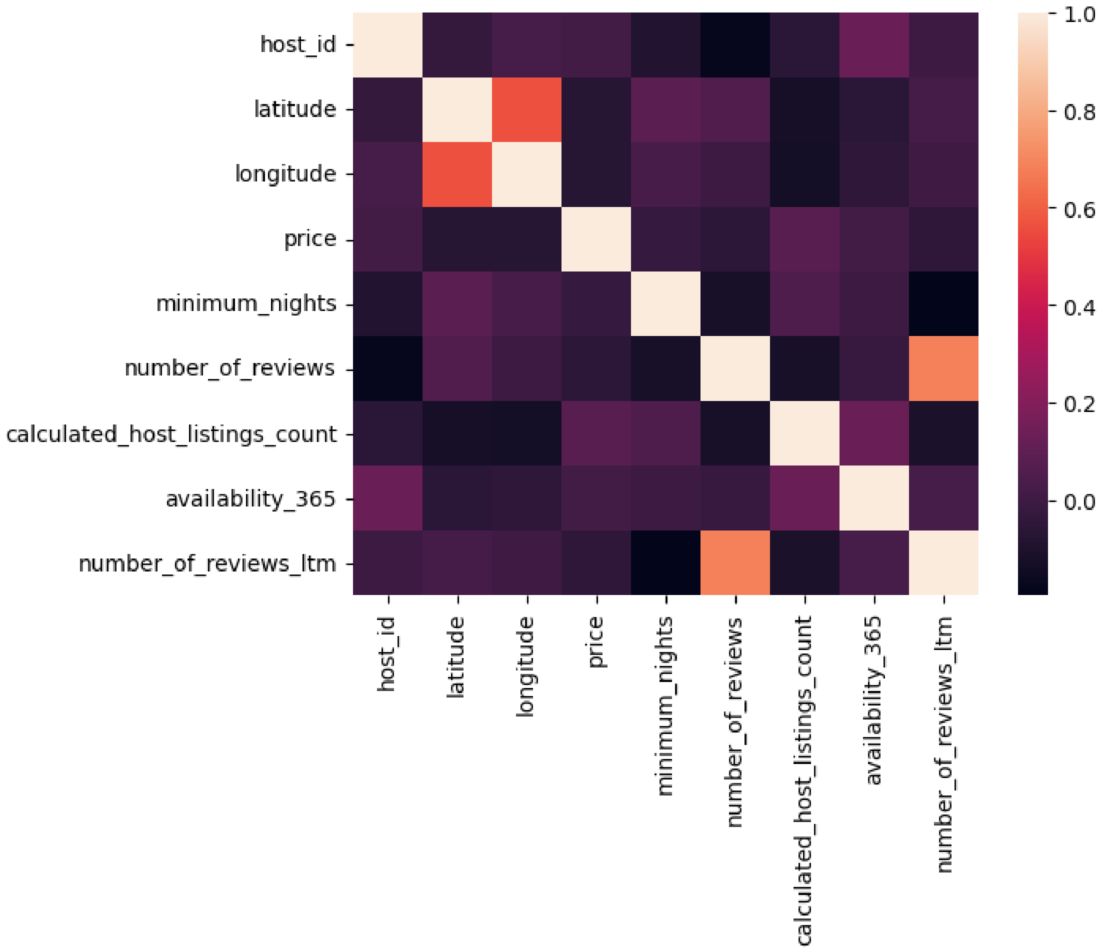


12. Create a matrix containing the correlations between the different columns in `df`. Plot it as a heatmap using Seaborn or similar. What does the plot tell you about correlations? Which columns are the most correlated to `price`?

```
[ ]: # Insert your code below
     # ======================


     corr = df.corr(numeric_only = True)
     sns.heatmap(corr)
```

```
[ ]: <AxesSubplot: >
```



The plot iss a heatmap, where the lighter the pixel is the higher the correlation is.

Price is best correlated with calculated_host_listing_count and slightly corelated with availability_365.