CSM148 Project 1

WILLIAM CULVER RANDALL

TOTAL POINTS

65 / 65

QUESTION 1

2. Visualizing Data 25 pts

1.1 2.0.1 Load the data + statistics **5 / 5**

- ✓ **- 0 pts** Correct
- **- 1 pts** display first 10 rows wrong
- **- 1 pts** drop colums wrong
- **- 1 pts** didn't provide a summary
- **- 1 pts** didn't plot histograms

1.2 2.0.2 Plot median price per neighbourhood_group **5 / 5**

- ✓ **- 0 pts** Correct
- **- 2.5 pts** didn't plot
- **- 5 pts** plot wrong diagram

1.3 2.0.3 Plot map of airbnbs throughout New York **5 / 5**

- ✓ **- 0 pts** Correct
- **- 1 pts** map scale not correct
- **- 5 pts** wrong plot map

1.4 2.0.4 Plot average price of hosts (host_id) who have more than 50 listings **5 / 5**

- ✓ **- 0 pts** Correct
- **- 2.5 pts** didn't plot
- **- 5 pts** plot wrong
- **- 2.5 pts** plot partially correct

1.5 2.0.5 Plot correlation matrix **5 / 5**

- ✓ **- 0 pts** Correct
- **- 1 pts** didn't plot

QUESTION 2

3. Prepare the Data 25 pts

2.1 3.0.1 Set aside 25% of the data as test set **5 / 5**

- ✓ **- 0 pts** Correct
- **- 5 pts** split wrong
- **- 5 pts** did not split

2.2 3.0.2 Augment the data frame with two other features which you think would be useful **5 / 5**

- ✓ **- 0 pts** Correct
- **- 5 pts** didn't augment data
- **- 2.5 pts** partially correct

2.3 3.0.3 Impute any missing feature with a method of your choice, and briefly discuss why you chose this imputation method **5 / 5**

- ✓ **- 0 pts** Correct
- **- 5 pts** did not impute missing feature
- **- 2.5 pts** did not provide reason

2.4 3.0.4 Code complete data pipeline using sklearn mixing **10 / 10**

- ✓ **- 0 pts** Correct
- **- 5 pts** partially correct
- **- 10 pts** didn't do data pipeline

QUESTION 3

3 4. Fit a model of your choice **15 / 15**

- ✓ **- 0 pts** Correct
- **- 2.5 pts** train MAE wrong or not provided. The MAE should be between 50-100ish for most cases (and not a very small number)
- **- 2.5 pts** test MAE wrong or not provided. The MAE

should be between 50-100ish for most cases (and not a very small number)

- 10 pts model fitting wrong

```
airbnb = pd.read_csv('datasets/airbnb/AB_NYC_2019.csv')
```

In [35]:

```
# display the first 10 rows of the data
airbnb.head(10)
```

Out[35]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_revie
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	2018-10-
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	Na
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-
5	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hill	40.74767	-73.97500	Entire home/apt	200	3	74	2019-06-
6	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40.68688	-73.95596	Private room	60	45	49	2017-10-
7	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Hell's Kitchen	40.76489	-73.98493	Private room	79	2	430	2019-06-
8	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Upper West Side	40.80178	-73.96723	Private room	79	2	118	2017-07-
9	5238	Cute & Cozy Lower East Side 1 bdrm	7549	Ben	Manhattan	Chinatown	40.71344	-73.99037	Entire home/apt	150	1	160	2019-06-

In [36]:

```
# drop the following columns: name, host_name, last_review
airbnb.drop(columns=["name", "host_name", "last_review"], axis=1, inplace=True)
# display a summary of the statistics of the loaded data
airbnb.info()
airbnb.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 13 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   host_id                         48895 non-null  int64
 2   neighbourhood_group             48895 non-null  object
 3   neighbourhood                   48895 non-null  object
 4   latitude                        48895 non-null  float64
 5   longitude                       48895 non-null  float64
 6   room_type                       48895 non-null  object
 7   price                           48895 non-null  int64
 8   minimum_nights                  48895 non-null  int64
 9   number_of_reviews               48895 non-null  int64
 10  reviews_per_month               38843 non-null  float64
 11  calculated_host_listings_count  48895 non-null  int64
 12  availability_365                48895 non-null  int64
dtypes: float64(3), int64(7), object(3)
memory usage: 4.8+ MB
```

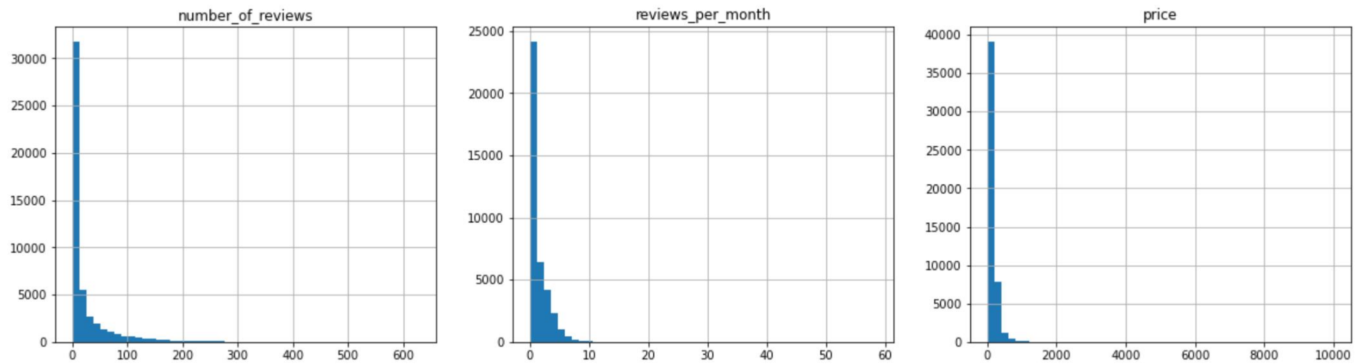
Out[36]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	a
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.143982	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.952519	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.000000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.000000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.000000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.000000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	

In [37]:

```
# plot histograms for 3 features of your choice
fig, (ax1,ax2,ax3) = plt.subplots(nrows=1, ncols=3,figsize=(20,5))
val1, val2, val3 = 'number_of_reviews', 'reviews_per_month', 'price'

ax1.title.set_text(val1)
airbnb[val1].hist(
    ax=ax1,
    bins=50)

ax2.title.set_text(val2)
airbnb[val2].hist(
    ax=ax2,
```

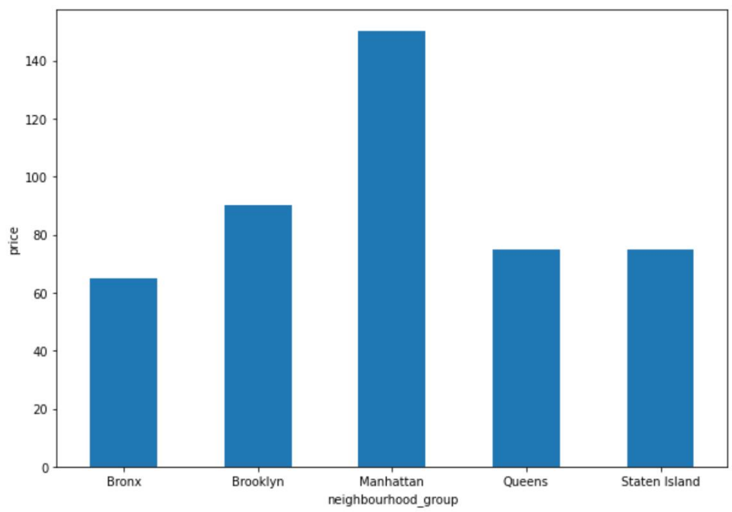
```
    bins=50)

ax3.title.set_text(val3)
airbnb[val3].hist(
    ax=ax3,
    bins=50)

plt.show()
```



[5 pts] Plot median price per neighbourhood_group

In [38]:
```
airbnb.groupby(['neighbourhood_group']).median()['price'].plot.bar(ylabel='price', rot=0, figsize=(10,7))
```

Out[38]: <AxesSubplot:xlabel='neighbourhood_group', ylabel='price'>



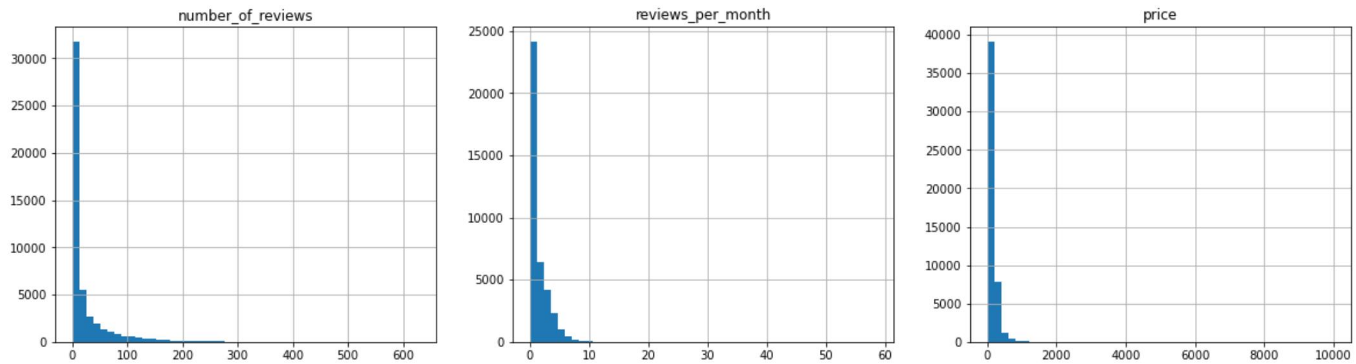
[5 pts] Plot map of airbnbs throughout New York (if it gets too crowded take a subset of the data, and try to make it look nice if you can :)).

In [39]:
```
x = "longitude"
y = "latitude"
ax = airbnb.plot(
    kind="scatter",
    x=x,
    y=y,
    figsize=(10,7),
    alpha=0.1)
plt.imshow(
    mpimg.imread('images/newyork.png'),
    extent = [-74.258, -73.7, 40.49,40.92],
    alpha=0.2)
plt.xlabel(x, fontsize=14)
plt.ylabel(y, fontsize=14)
plt.show()
```

1.1 2.0.1 Load the data + statistics **5 / 5**

- ✓ **- 0 pts** **Correct**
- **- 1 pts** display first 10 rows wrong
- **- 1 pts** drop colums wrong
- **- 1 pts** didn't provide a summary
- **- 1 pts** didn't plot histograms

```
    bins=50)

ax3.title.set_text(val3)
airbnb[val3].hist(
    ax=ax3,
    bins=50)

plt.show()
```



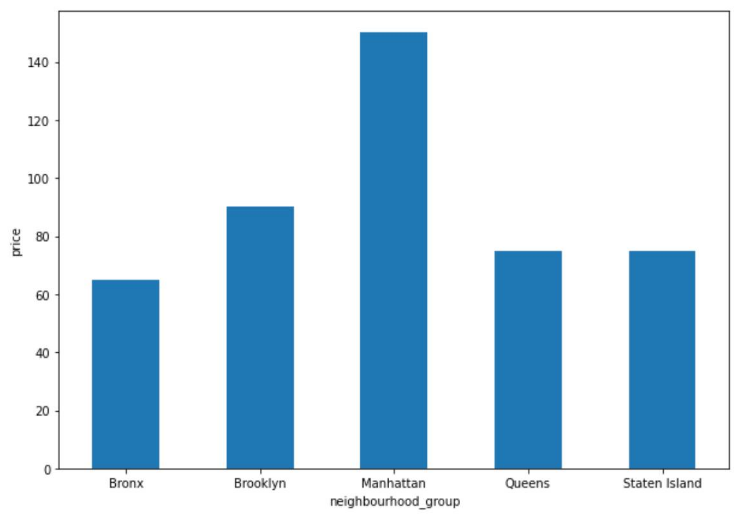
[5 pts] Plot median price per neighbourhood_group

In [38]:

```
airbnb.groupby(['neighbourhood_group']).median()['price'].plot.bar(ylabel='price', rot=0, figsize=(10,7))
```

Out[38]:

```
<AxesSubplot:xlabel='neighbourhood_group', ylabel='price'>
```



[5 pts] Plot map of airbnbs throughout New York (if it gets too crowded take a subset of the data, and try to make it look nice if you can :)).

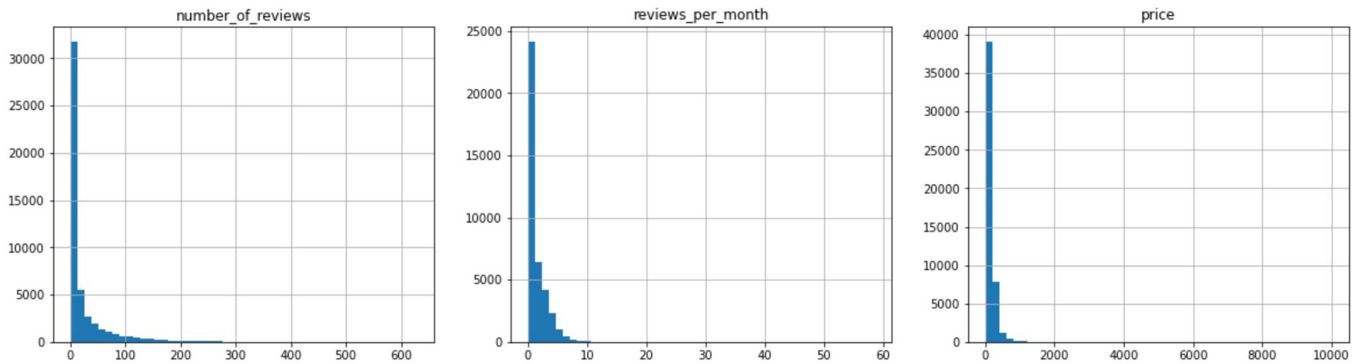
In [39]:

```
x = "longitude"
y = "latitude"
ax = airbnb.plot(
    kind="scatter",
    x=x,
    y=y,
    figsize=(10,7),
    alpha=0.1)
plt.imshow(
    mpimg.imread('images/newyork.png'),
    extent = [-74.258, -73.7, 40.49,40.92],
    alpha=0.2)
plt.xlabel(x, fontsize=14)
plt.ylabel(y, fontsize=14)
plt.show()
```

1.2 2.0.2 Plot median price per neighbourhood_group 5 / 5

- ✓ **- 0 pts** Correct
- **- 2.5 pts** didn't plot
- **- 5 pts** plot wrong diagram

```
    bins=50)

ax3.title.set_text(val3)
airbnb[val3].hist(
    ax=ax3,
    bins=50)

plt.show()
```



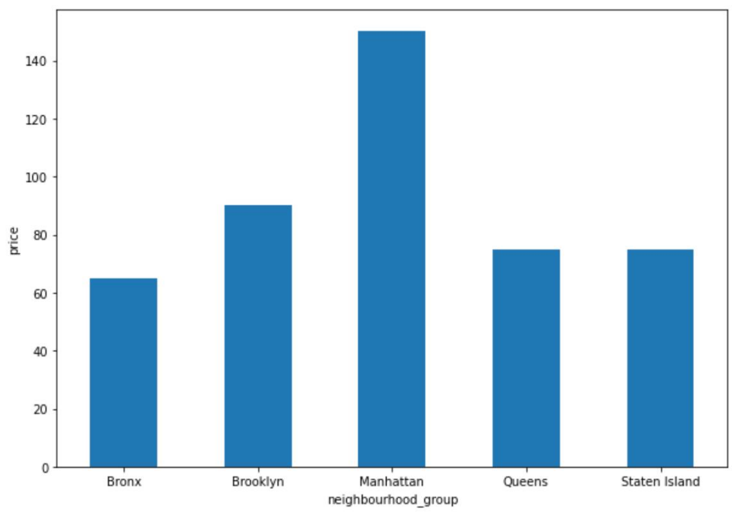
[5 pts] Plot median price per neighbourhood_group

In [38]:

```
airbnb.groupby(['neighbourhood_group']).median()['price'].plot.bar(ylabel='price', rot=0, figsize=(10,7))
```

Out[38]:

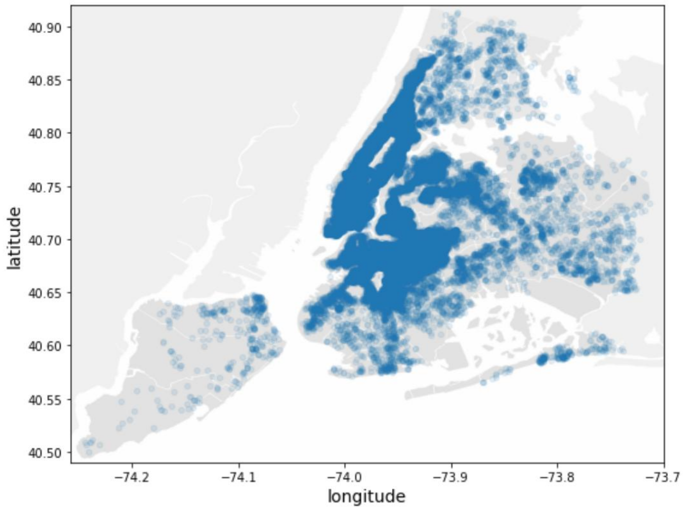
```
<AxesSubplot:xlabel='neighbourhood_group', ylabel='price'>
```



[5 pts] Plot map of airbnbs throughout New York (if it gets too crowded take a subset of the data, and try to make it look nice if you can :)).

In [39]:

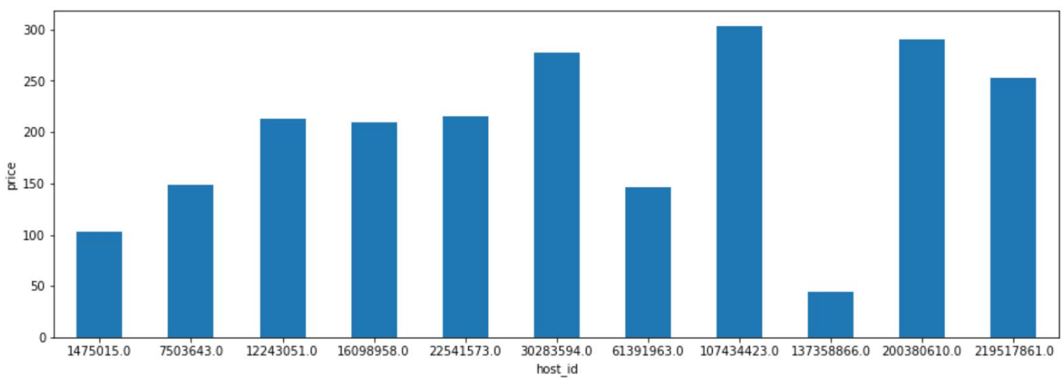
```
x = "longitude"
y = "latitude"
ax = airbnb.plot(
    kind="scatter",
    x=x,
    y=y,
    figsize=(10,7),
    alpha=0.1)
plt.imshow(
    mpimg.imread('images/newyork.png'),
    extent = [-74.258, -73.7, 40.49,40.92],
    alpha=0.2)
plt.xlabel(x, fontsize=14)
plt.ylabel(y, fontsize=14)
plt.show()
```



[5 pts] Plot average price of hosts (host_id) who have more than 50 listings.

In [40]:
```
airbnb.where(airbnb['calculated_host_listings_count'] > 50) \
    .groupby('host_id') \
    .mean('price')['price'] \
    .plot.bar(ylabel='price', figsize=(15,5), rot=0)
```

Out[40]: <AxesSubplot:xlabel='host_id', ylabel='price'>



[5 pts] Plot correlation matrix

- which features have positive correlation?
  - `reviews_per_month` and `number_of_reviews` have a strong positive correlation.
  - `availability_365` and `minimum_nights` have a slight positive correlation
  - `calculated_host_listings_count` and `minimum_nights` have a slight positive correlation
- which features have negative correlation?
  - `longitude` and `price` have a very slight negative correlation.
  - `reviews_per_month` and `minimum_nights` have a slight negative correlation
  - `longitude` and `calculated_host_listings_count` have a small negative correlation

Overall the correlation plot gives us very little information about the interconnectedness of the attributes.

In [41]:
```
corr_matrix = airbnb.corr()
```

In [42]:
```
attributes = [
    'price',
    'longitude',
    'latitude',
    'minimum_nights',
    'number_of_reviews',
    'reviews_per_month',
    'calculated_host_listings_count',
    'availability_365']
for attribute in attributes:
    print(f'{attribute}\n{corr_matrix[attribute].sort_values(ascending=False)}\n')
```

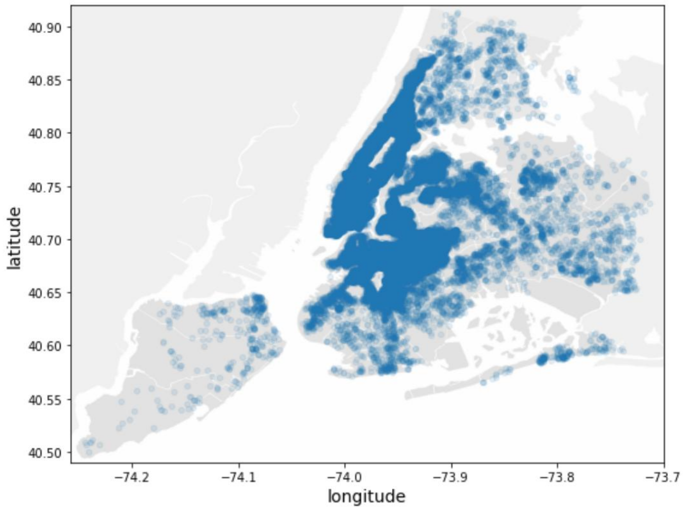
```
price
price                             1.000000
availability_365                  0.081829
calculated_host_listings_count    0.057472
minimum_nights                    0.042799
latitude                          0.033939
host_id                           0.015309
id                                0.010619
reviews_per_month                -0.030608
number_of_reviews                -0.047954
```

1.3 2.0.3 Plot map of airbnbs throughout New York **5 / 5**

✓ **- 0 pts** **Correct**

- 1 pts map scale not correct

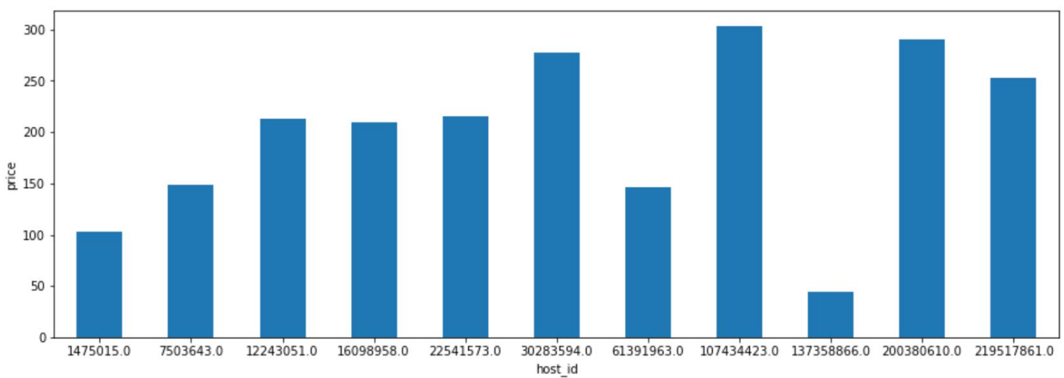
- 5 pts wrong plot map



[5 pts] Plot average price of hosts (host_id) who have more than 50 listings.

In [40]:
```
airbnb.where(airbnb['calculated_host_listings_count'] > 50) \
    .groupby('host_id') \
    .mean('price')['price'] \
    .plot.bar(ylabel='price', figsize=(15,5), rot=0)
```

Out[40]: <AxesSubplot:xlabel='host_id', ylabel='price'>



[5 pts] Plot correlation matrix

- which features have positive correlation?
  - `reviews_per_month` and `number_of_reviews` have a strong positive correlation.
  - `availability_365` and `minimum_nights` have a slight positive correlation
  - `calculated_host_listings_count` and `minimum_nights` have a slight positive correlation
- which features have negative correlation?
  - `longitude` and `price` have a very slight negative correlation.
  - `reviews_per_month` and `minimum_nights` have a slight negative correlation
  - `longitude` and `calculated_host_listings_count` have a small negative correlation

Overall the correlation plot gives us very little information about the interconnectedness of the attributes.

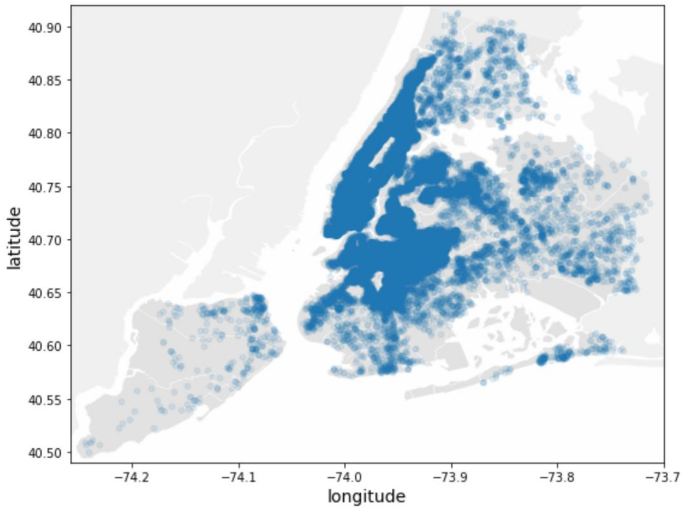
In [41]:
```
corr_matrix = airbnb.corr()
```

In [42]:
```
attributes = [
    'price',
    'longitude',
    'latitude',
    'minimum_nights',
    'number_of_reviews',
    'reviews_per_month',
    'calculated_host_listings_count',
    'availability_365']
for attribute in attributes:
    print(f'{attribute}\n{corr_matrix[attribute].sort_values(ascending=False)}\n')
```

```
price
price                             1.000000
availability_365                  0.081829
calculated_host_listings_count    0.057472
minimum_nights                    0.042799
latitude                          0.033939
host_id                           0.015309
id                                0.010619
reviews_per_month                -0.030608
number_of_reviews                -0.047954
```

1.4 2.0.4 Plot average price of hosts (host_id) who have more than 50 listings **5 / 5**

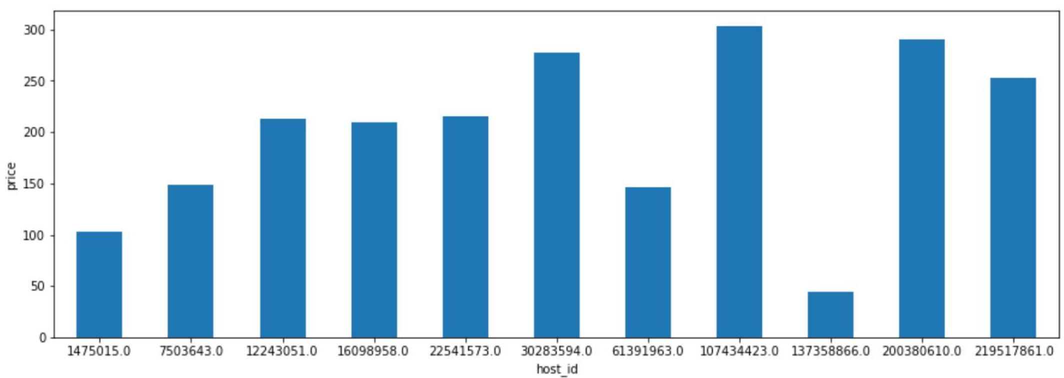
- ✓ **- 0 pts** Correct
- **- 2.5 pts** didn't plot
- **- 5 pts** plot wrong
- **- 2.5 pts** plot partially correct



[5 pts] Plot average price of hosts (host_id) who have more than 50 listings.

In [40]:
```
airbnb.where(airbnb['calculated_host_listings_count'] > 50) \
    .groupby('host_id') \
    .mean('price')['price'] \
    .plot.bar(ylabel='price', figsize=(15,5), rot=0)
```

Out[40]: <AxesSubplot:xlabel='host_id', ylabel='price'>



[5 pts] Plot correlation matrix

- which features have positive correlation?
  - `reviews_per_month` and `number_of_reviews` have a strong positive correlation.
  - `availability_365` and `minimum_nights` have a slight positive correlation
  - `calculated_host_listings_count` and `minimum_nights` have a slight positive correlation
- which features have negative correlation?
  - `longitude` and `price` have a very slight negative correlation.
  - `reviews_per_month` and `minimum_nights` have a slight negative correlation
  - `longitude` and `calculated_host_listings_count` have a small negative correlation

Overall the correlation plot gives us very little information about the interconnectedness of the attributes.

In [41]:
```
corr_matrix = airbnb.corr()
```

In [42]:
```
attributes = [
    'price',
    'longitude',
    'latitude',
    'minimum_nights',
    'number_of_reviews',
    'reviews_per_month',
    'calculated_host_listings_count',
    'availability_365']
for attribute in attributes:
    print(f'{attribute}\n{corr_matrix[attribute].sort_values(ascending=False)}\n')
```

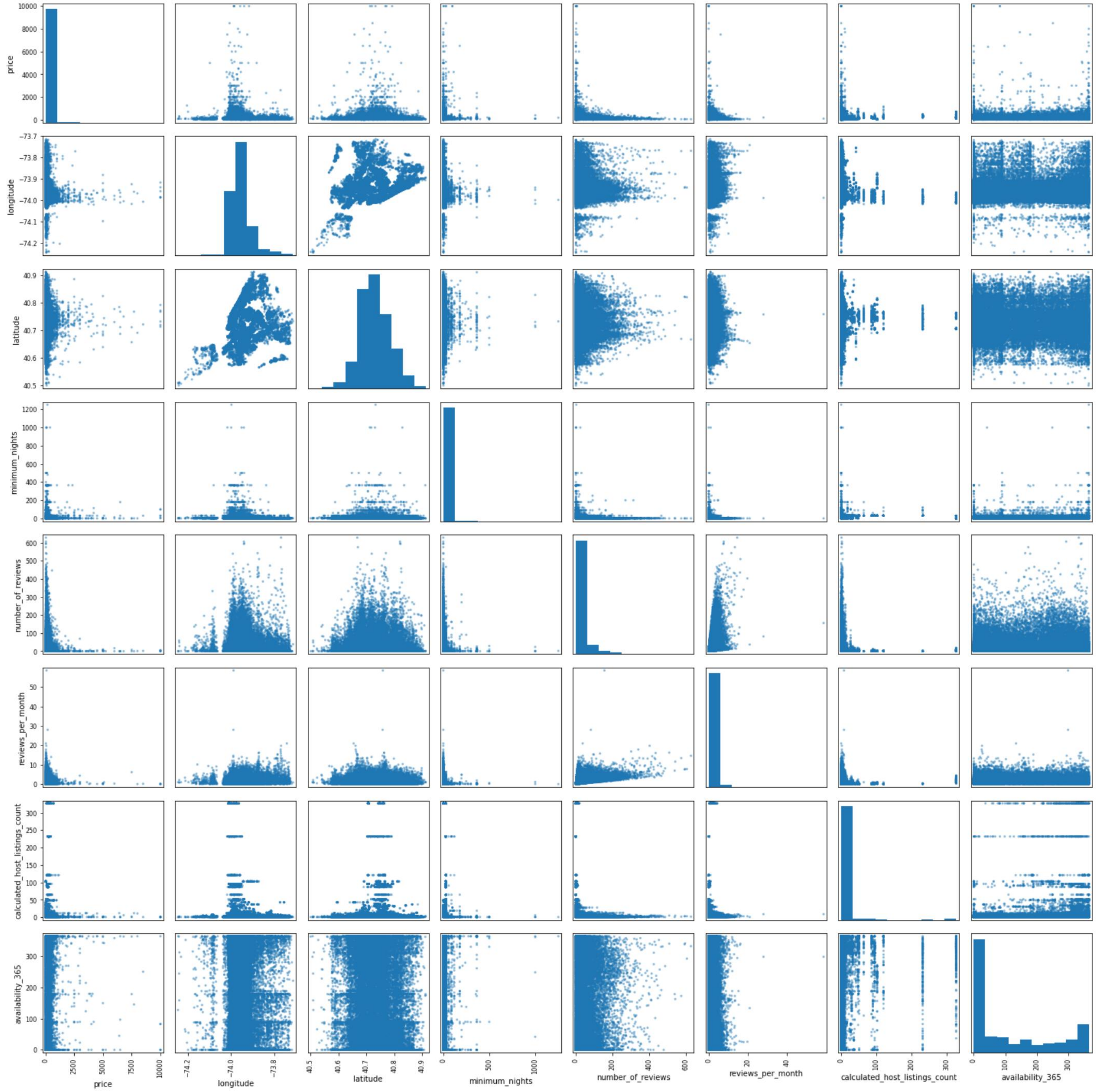
```
price
price                             1.000000
availability_365                  0.081829
calculated_host_listings_count    0.057472
minimum_nights                    0.042799
latitude                          0.033939
host_id                           0.015309
id                                0.010619
reviews_per_month                -0.030608
number_of_reviews                -0.047954
```

```
longitude                        -0.150019
Name: price, dtype: float64

longitude
longitude                         1.000000
reviews_per_month                 0.145948
host_id                           0.127055
id                                0.090908
latitude                          0.084788
availability_365                  0.082731
number_of_reviews                 0.059094
minimum_nights                   -0.062747
calculated_host_listings_count   -0.114713
price                            -0.150019
Name: longitude, dtype: float64

latitude
latitude                          1.000000
longitude                         0.084788
price                             0.033939
minimum_nights                    0.024869
host_id                           0.020224
calculated_host_listings_count    0.019517
id                               -0.003125
reviews_per_month                -0.010142
availability_365                 -0.010983
number_of_reviews                -0.015389
Name: latitude, dtype: float64

minimum_nights
minimum_nights                    1.000000
availability_365                  0.144303
calculated_host_listings_count    0.127960
price                             0.042799
latitude                          0.024869
id                               -0.013224
host_id                          -0.017364
longitude                        -0.062747
number_of_reviews                -0.080116
reviews_per_month                -0.121702
Name: minimum_nights, dtype: float64

number_of_reviews
number_of_reviews                 1.000000
reviews_per_month                 0.549868
availability_365                  0.172028
longitude                         0.059094
latitude                         -0.015389
price                            -0.047954
calculated_host_listings_count   -0.072376
minimum_nights                   -0.080116
host_id                          -0.140106
id                               -0.319760
Name: number_of_reviews, dtype: float64

reviews_per_month
reviews_per_month                 1.000000
number_of_reviews                 0.549868
host_id                           0.296417
id                                0.291828
availability_365                  0.185791
longitude                         0.145948
calculated_host_listings_count   -0.009421
latitude                         -0.010142
price                            -0.030608
minimum_nights                   -0.121702
Name: reviews_per_month, dtype: float64

calculated_host_listings_count
calculated_host_listings_count    1.000000
availability_365                  0.225701
host_id                           0.154950
id                                0.133272
minimum_nights                    0.127960
price                             0.057472
latitude                          0.019517
reviews_per_month                -0.009421
number_of_reviews                -0.072376
longitude                        -0.114713
Name: calculated_host_listings_count, dtype: float64

availability_365
availability_365                  1.000000
calculated_host_listings_count    0.225701
host_id                           0.203492
reviews_per_month                 0.185791
number_of_reviews                 0.172028
minimum_nights                    0.144303
id                                0.085468
longitude                         0.082731
price                             0.081829
latitude                         -0.010983
Name: availability_365, dtype: float64
```

In [43]:

```
scatter_matrix(airbnb[attributes], figsize=(20,20))
save_fig('correlation')
plt.show()
```

```
Saving figure correlation
```



[25 pts] Prepare the Data

[5 pts] Set aside 25% of the data as test set (75% train, 25% test).

In [44]:
```
airbnb.describe()
```

Out[44]:

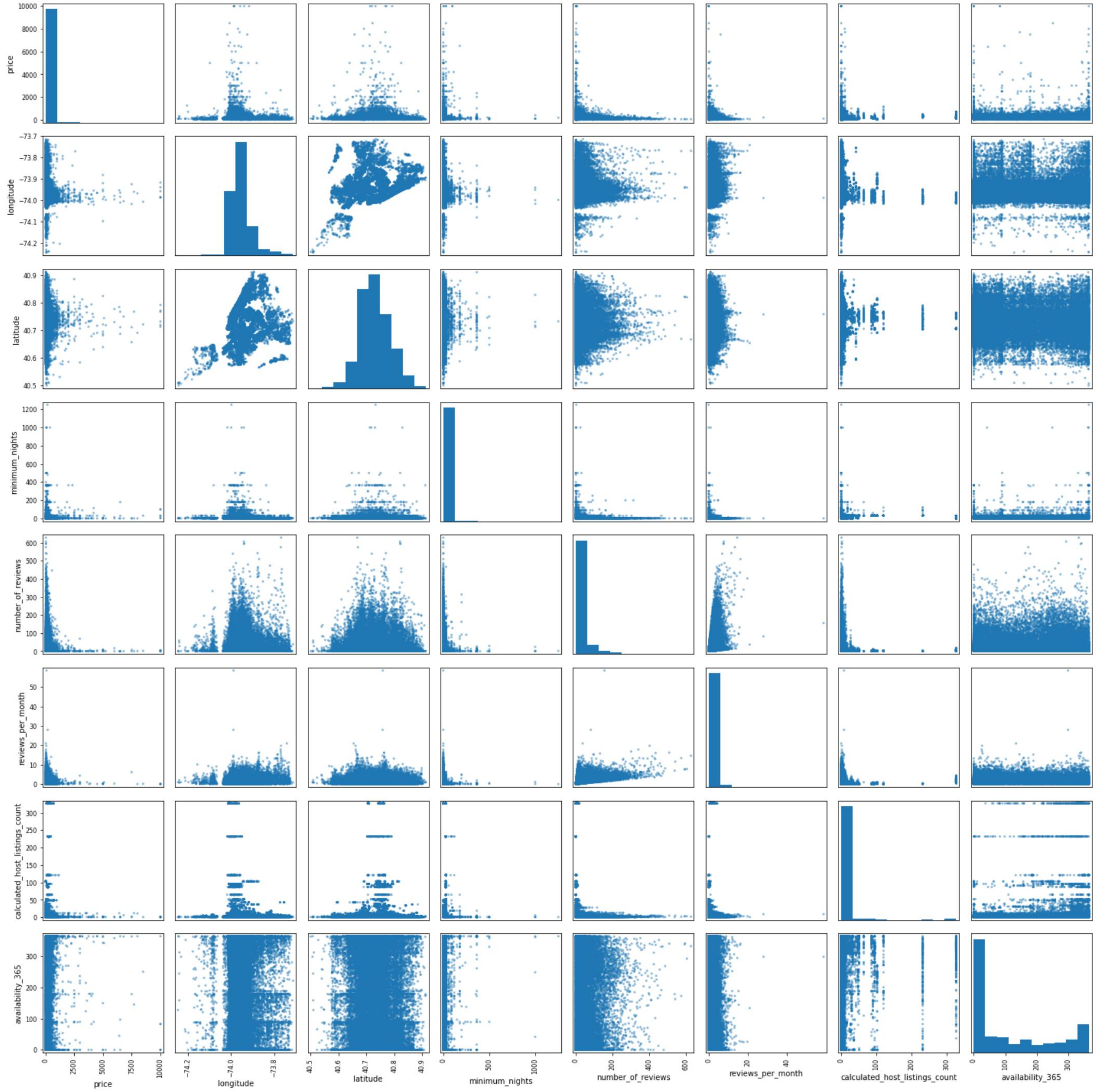
	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	a
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.143982	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.952519	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.000000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.000000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.000000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.000000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	

In [45]:
```
price = 'price'
categorical_price = 'categorical_price'
```

1.5 2.0.5 Plot correlation matrix 5 / 5

✓ **- 0 pts** Correct

- 1 pts didn't plot



[25 pts] Prepare the Data

[5 pts] Set aside 25% of the data as test set (75% train, 25% test).

In [44]:

```
airbnb.describe()
```

Out[44]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	a
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.143982	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.952519	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.000000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.000000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.000000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.000000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	

In [45]:

```
price = 'price'
categorical_price = 'categorical_price'
```

```python
bins = [-1,100.0, 200.0, 300.0, 400.0, 500.0, 600.0, 700.0, 800.0, 900.0, 1000.0, 2500.0, 5000.0, 10000.0, np.inf]
for i in range(1,len(bins)):
    assert(bins[i]-bins[i-1]>0)
airbnb[categorical_price] = pd.cut(airbnb[price],bins = bins,labels = [i for i in range(1, len(bins))])

sss = StratifiedShuffleSplit(n_splits=1, test_size=0.25, random_state=42)
for train_index, test_index in sss.split(airbnb, airbnb[categorical_price]):
    abnb_train_set = airbnb.loc[train_index]
    abnb_test_set = airbnb.loc[test_index]


# labels are actual price
abnb_test_labels = abnb_test_set[price].copy()
abnb_train_labels = abnb_train_set[price].copy()

# train set should not contain actual price
abnb_train_set.drop(price, axis=1, inplace=True)

# test set should not include price
abnb_test_set.drop(price, axis=1, inplace=True)
```

In [46]:
```python
abnb_train_set.describe()
```

Out[46]:

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	3.667100e+04	3.667100e+04	36671.000000	36671.000000	36671.000000	36671.000000	29220.000000	36671.000000	36671.000000
mean	1.895560e+07	6.741661e+07	40.729026	-73.952103	7.022715	23.416242	1.368056	7.156309	112.870770
std	1.097103e+07	7.855348e+07	0.054523	0.046339	20.777483	44.880430	1.687344	33.079026	131.655161
min	3.647000e+03	2.438000e+03	40.499790	-74.244420	1.000000	0.000000	0.010000	1.000000	0.000000
25%	9.422985e+06	7.737249e+06	40.690160	-73.983030	1.000000	1.000000	0.190000	1.000000	0.000000
50%	1.960362e+07	3.037019e+07	40.723200	-73.955650	2.000000	5.000000	0.710000	1.000000	46.000000
75%	2.905732e+07	1.074344e+08	40.763155	-73.936150	5.000000	24.000000	2.020000	2.000000	227.000000
max	3.648724e+07	2.743115e+08	40.913060	-73.716900	1250.000000	629.000000	58.500000	327.000000	365.000000

In [47]:
```python
abnb_test_set.describe()
```

Out[47]:

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	1.222400e+04	1.222400e+04	12224.000000	12224.000000	12224.000000	12224.000000	9623.000000	12224.000000	12224.000000
mean	1.920178e+07	6.823019e+07	40.728718	-73.952369	7.051702	22.849149	1.388907	7.107003	112.513007
std	1.101764e+07	7.878324e+07	0.054552	0.045606	19.688882	43.545117	1.659298	32.571374	131.528644
min	2.539000e+03	2.787000e+03	40.522110	-74.198260	1.000000	0.000000	0.010000	1.000000	0.000000
25%	9.606994e+06	8.055524e+06	40.689880	-73.983190	1.000000	1.000000	0.200000	1.000000	0.000000
50%	1.988499e+07	3.216326e+07	40.722540	-73.955875	3.000000	5.000000	0.740000	1.000000	43.000000
75%	2.946058e+07	1.074344e+08	40.763042	-73.936690	5.000000	23.000000	2.020000	2.000000	225.000000
max	3.648561e+07	2.743213e+08	40.911670	-73.712990	999.000000	576.000000	17.820000	327.000000	365.000000

In [48]:
```python
abnb_test_labels, abnb_train_labels
```

Out[48]:
```
(8048      75
 16686    175
 32367    160
 35840     80
 9220     155
         ...
 34916    170
 45515    499
 46000    650
 48801    155
 44136     70
 Name: price, Length: 12224, dtype: int64,
 5822      80
 24749    150
 6081     200
 9623      40
 4713      44
         ...
 4613      55
 24840     65
 22868     40
 48563    343
 32423     69
 Name: price, Length: 36671, dtype: int64)
```

[5 pts] Augment the dataframe with two other features which you think would be useful

In [49]:
```python
# months since they have been posting
num_months_posting = 'num_months_posting'
number_of_reviews = 'number_of_reviews'
reviews_per_month = 'reviews_per_month'
abnb_train_set[num_months_posting] = abnb_train_set[number_of_reviews] / abnb_train_set[reviews_per_month]
abnb_test_set[num_months_posting] = abnb_test_set[number_of_reviews] / abnb_test_set[reviews_per_month]
abnb_train_set[num_months_posting].fillna(0,inplace=True)
abnb_test_set[num_months_posting].fillna(0,inplace=True)

# how much of the year does a single booking take up
```

2.1 3.0.1 Set aside 25% of the data as test set 5 / 5

✓ **- 0 pts** Correct

- 5 pts split wrong

- 5 pts did not split

```
bins = [-1,100.0, 200.0, 300.0, 400.0, 500.0, 600.0, 700.0, 800.0, 900.0, 1000.0, 2500.0, 5000.0, 10000.0, np.inf]
for i in range(1,len(bins)):
    assert(bins[i]-bins[i-1]>0)
airbnb[categorical_price] = pd.cut(airbnb[price],bins = bins,labels = [i for i in range(1, len(bins))])

sss = StratifiedShuffleSplit(n_splits=1, test_size=0.25, random_state=42)
for train_index, test_index in sss.split(airbnb, airbnb[categorical_price]):
    abnb_train_set = airbnb.loc[train_index]
    abnb_test_set = airbnb.loc[test_index]


# labels are actual price
abnb_test_labels = abnb_test_set[price].copy()
abnb_train_labels = abnb_train_set[price].copy()

# train set should not contain actual price
abnb_train_set.drop(price, axis=1, inplace=True)

# test set should not include price
abnb_test_set.drop(price, axis=1, inplace=True)
```

In [46]:

```
abnb_train_set.describe()
```

Out[46]:

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	3.667100e+04	3.667100e+04	36671.000000	36671.000000	36671.000000	36671.000000	29220.000000	36671.000000	36671.000000
mean	1.895560e+07	6.741661e+07	40.729026	-73.952103	7.022715	23.416242	1.368056	7.156309	112.870770
std	1.097103e+07	7.855348e+07	0.054523	0.046339	20.777483	44.880430	1.687344	33.079026	131.655161
min	3.647000e+03	2.438000e+03	40.499790	-74.244420	1.000000	0.000000	0.010000	1.000000	0.000000
25%	9.422985e+06	7.737249e+06	40.690160	-73.983030	1.000000	1.000000	0.190000	1.000000	0.000000
50%	1.960362e+07	3.037019e+07	40.723200	-73.955650	2.000000	5.000000	0.710000	1.000000	46.000000
75%	2.905732e+07	1.074344e+08	40.763155	-73.936150	5.000000	24.000000	2.020000	2.000000	227.000000
max	3.648724e+07	2.743115e+08	40.913060	-73.716900	1250.000000	629.000000	58.500000	327.000000	365.000000

In [47]:

```
abnb_test_set.describe()
```

Out[47]:

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	1.222400e+04	1.222400e+04	12224.000000	12224.000000	12224.000000	12224.000000	9623.000000	12224.000000	12224.000000
mean	1.920178e+07	6.823019e+07	40.728718	-73.952369	7.051702	22.849149	1.388907	7.107003	112.513007
std	1.101764e+07	7.878324e+07	0.054552	0.045606	19.688882	43.545117	1.659298	32.571374	131.528644
min	2.539000e+03	2.787000e+03	40.522110	-74.198260	1.000000	0.000000	0.010000	1.000000	0.000000
25%	9.606994e+06	8.055524e+06	40.689880	-73.983190	1.000000	1.000000	0.200000	1.000000	0.000000
50%	1.988499e+07	3.216326e+07	40.722540	-73.955875	3.000000	5.000000	0.740000	1.000000	43.000000
75%	2.946058e+07	1.074344e+08	40.763042	-73.936690	5.000000	23.000000	2.020000	2.000000	225.000000
max	3.648561e+07	2.743213e+08	40.911670	-73.712990	999.000000	576.000000	17.820000	327.000000	365.000000

In [48]:

```
abnb_test_labels, abnb_train_labels
```

Out[48]:

```
(8048      75
 16686    175
 32367    160
 35840     80
 9220     155
         ...
 34916    170
 45515    499
 46000    650
 48801    155
 44136     70
 Name: price, Length: 12224, dtype: int64,
 5822      80
 24749    150
 6081     200
 9623      40
 4713      44
         ...
 4613      55
 24840     65
 22868     40
 48563    343
 32423     69
 Name: price, Length: 36671, dtype: int64)
```

[5 pts] Augment the dataframe with two other features which you think would be useful

In [49]:

```
# months since they have been posting
num_months_posting = 'num_months_posting'
number_of_reviews = 'number_of_reviews'
reviews_per_month = 'reviews_per_month'
abnb_train_set[num_months_posting] = abnb_train_set[number_of_reviews] / abnb_train_set[reviews_per_month]
abnb_test_set[num_months_posting] = abnb_test_set[number_of_reviews] / abnb_test_set[reviews_per_month]
abnb_train_set[num_months_posting].fillna(0,inplace=True)
abnb_test_set[num_months_posting].fillna(0,inplace=True)

# how much of the year does a single booking take up
```

```
one_stay_percent_of_year = 'one_stay_percent_of_year'
availability_365 = 'availability_365'
minimum_nights = 'minimum_nights'
abnb_train_set[one_stay_percent_of_year] = abnb_train_set[minimum_nights] / abnb_train_set[availability_365]
abnb_test_set[one_stay_percent_of_year] = abnb_test_set[minimum_nights] / abnb_test_set[availability_365]
```

[5 pts] Impute any missing feature with a method of your choice, and briefly discuss why you chose this imputation method

In [50]:

```
# ANSWER
# nan reviews per month => assume median number of reviews
# If there is a NaN for the number of reviews per month I will set it to the medain of the whole dataset
reviews_per_month = 'reviews_per_month'
median_reviews_per_month = airbnb[reviews_per_month].median()
abnb_train_set[reviews_per_month].fillna(median_reviews_per_month,inplace=True)
abnb_test_set[reviews_per_month].fillna(median_reviews_per_month,inplace=True)
```

In [51]:

```
airbnb.values
```

Out[51]:

```
array([[2539, 2787, 'Brooklyn', ..., 6, 365, 2],
       [2595, 2845, 'Manhattan', ..., 2, 355, 3],
       [3647, 4632, 'Manhattan', ..., 1, 365, 2],
       ...,
       [36485431, 23492952, 'Manhattan', ..., 1, 27, 2],
       [36485609, 30985759, 'Manhattan', ..., 6, 2, 1],
       [36487245, 68119814, 'Manhattan', ..., 1, 23, 1]], dtype=object)
```

In [52]:

```
airbnb.head()
```

Out[52]:

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_li
0	2539	2787	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	0.21	
1	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	0.38	
2	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	
3	3831	4869	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	4.64	
4	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	0.10	

In [53]:

```
airbnb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   host_id                         48895 non-null  int64
 2   neighbourhood_group             48895 non-null  object
 3   neighbourhood                   48895 non-null  object
 4   latitude                        48895 non-null  float64
 5   longitude                       48895 non-null  float64
 6   room_type                       48895 non-null  object
 7   price                           48895 non-null  int64
 8   minimum_nights                  48895 non-null  int64
 9   number_of_reviews               48895 non-null  int64
 10  reviews_per_month               38843 non-null  float64
 11  calculated_host_listings_count  48895 non-null  int64
 12  availability_365                48895 non-null  int64
 13  categorical_price               48895 non-null  category
dtypes: category(1), float64(3), int64(7), object(3)
memory usage: 4.9+ MB
```

[10 pts] Code complete data pipeline using sklearn mixins

In [54]:

```
from sklearn.model_selection import train_test_split
airbnb = pd.read_csv('datasets/airbnb/AB_NYC_2019.csv')
airbnb_true_vals = airbnb['price'].copy()
airbnb.drop(columns=["name", "host_name", "last_review","id", "host_id", "price"], axis=1, inplace=True)
cat_feat = ['neighbourhood_group', 'neighbourhood', 'room_type']
numeric_columns = airbnb.drop(columns=cat_feat,axis=1)
numeric_features = list(numeric_columns)
print(f'{numeric_features=}')

print(airbnb.info())
```

```
numeric_features=['latitude', 'longitude', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availa
bility_365']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   neighbourhood_group             48895 non-null  object
 1   neighbourhood                   48895 non-null  object
 2   latitude                        48895 non-null  float64
 3   longitude                       48895 non-null  float64
 4   room_type                       48895 non-null  object
 5   minimum_nights                  48895 non-null  int64
 6   number_of_reviews               48895 non-null  int64
```

2.2 3.0.2 Augment the data frame with two other features which you think would be useful 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts didn't augment data

- 2.5 pts partially correct

```
one_stay_percent_of_year = 'one_stay_percent_of_year'
availability_365 = 'availability_365'
minimum_nights = 'minimum_nights'
abnb_train_set[one_stay_percent_of_year] = abnb_train_set[minimum_nights] / abnb_train_set[availability_365]
abnb_test_set[one_stay_percent_of_year] = abnb_test_set[minimum_nights] / abnb_test_set[availability_365]
```

[5 pts] Impute any missing feature with a method of your choice, and briefly discuss why you chose this imputation method

In [50]:

```
# ANSWER
# nan reviews per month => assume median number of reviews
# If there is a NaN for the number of reviews per month I will set it to the medain of the whole dataset
reviews_per_month = 'reviews_per_month'
median_reviews_per_month = airbnb[reviews_per_month].median()
abnb_train_set[reviews_per_month].fillna(median_reviews_per_month,inplace=True)
abnb_test_set[reviews_per_month].fillna(median_reviews_per_month,inplace=True)
```

In [51]:

```
airbnb.values
```

Out[51]:

```
array([[2539, 2787, 'Brooklyn', ..., 6, 365, 2],
       [2595, 2845, 'Manhattan', ..., 2, 355, 3],
       [3647, 4632, 'Manhattan', ..., 1, 365, 2],
       ...,
       [36485431, 23492952, 'Manhattan', ..., 1, 27, 2],
       [36485609, 30985759, 'Manhattan', ..., 6, 2, 1],
       [36487245, 68119814, 'Manhattan', ..., 1, 23, 1]], dtype=object)
```

In [52]:

```
airbnb.head()
```

Out[52]:

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_li
0	2539	2787	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	0.21	
1	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	0.38	
2	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	
3	3831	4869	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	4.64	
4	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	0.10	

In [53]:

```
airbnb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   host_id                         48895 non-null  int64
 2   neighbourhood_group             48895 non-null  object
 3   neighbourhood                   48895 non-null  object
 4   latitude                        48895 non-null  float64
 5   longitude                       48895 non-null  float64
 6   room_type                       48895 non-null  object
 7   price                           48895 non-null  int64
 8   minimum_nights                  48895 non-null  int64
 9   number_of_reviews               48895 non-null  int64
 10  reviews_per_month               38843 non-null  float64
 11  calculated_host_listings_count  48895 non-null  int64
 12  availability_365                48895 non-null  int64
 13  categorical_price               48895 non-null  category
dtypes: category(1), float64(3), int64(7), object(3)
memory usage: 4.9+ MB
```

[10 pts] Code complete data pipeline using sklearn mixins

In [54]:

```
from sklearn.model_selection import train_test_split
airbnb = pd.read_csv('datasets/airbnb/AB_NYC_2019.csv')
airbnb_true_vals = airbnb['price'].copy()
airbnb.drop(columns=["name", "host_name", "last_review","id", "host_id", "price"], axis=1, inplace=True)
cat_feat = ['neighbourhood_group', 'neighbourhood', 'room_type']
numeric_columns = airbnb.drop(columns=cat_feat,axis=1)
numeric_features = list(numeric_columns)
print(f'{numeric_features=}')

print(airbnb.info())
```

```
numeric_features=['latitude', 'longitude', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availa
bility_365']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   neighbourhood_group             48895 non-null  object
 1   neighbourhood                   48895 non-null  object
 2   latitude                        48895 non-null  float64
 3   longitude                       48895 non-null  float64
 4   room_type                       48895 non-null  object
 5   minimum_nights                  48895 non-null  int64
 6   number_of_reviews               48895 non-null  int64
```

2.3 3.0.3 Impute any missing feature with a method of your choice, and briefly discuss why you chose this imputation method **5 / 5**

- ✓ **- 0 pts** **Correct**
- **- 5 pts** did not impute missing feature
- **- 2.5 pts** did not provide reason

```
one_stay_percent_of_year = 'one_stay_percent_of_year'
availability_365 = 'availability_365'
minimum_nights = 'minimum_nights'
abnb_train_set[one_stay_percent_of_year] = abnb_train_set[minimum_nights] / abnb_train_set[availability_365]
abnb_test_set[one_stay_percent_of_year] = abnb_test_set[minimum_nights] / abnb_test_set[availability_365]
```

[5 pts] Impute any missing feature with a method of your choice, and briefly discuss why you chose this imputation method

In [50]:

```
# ANSWER
# nan reviews per month => assume median number of reviews
# If there is a NaN for the number of reviews per month I will set it to the medain of the whole dataset
reviews_per_month = 'reviews_per_month'
median_reviews_per_month = airbnb[reviews_per_month].median()
abnb_train_set[reviews_per_month].fillna(median_reviews_per_month,inplace=True)
abnb_test_set[reviews_per_month].fillna(median_reviews_per_month,inplace=True)
```

In [51]:

```
airbnb.values
```

Out[51]:

```
array([[2539, 2787, 'Brooklyn', ..., 6, 365, 2],
       [2595, 2845, 'Manhattan', ..., 2, 355, 3],
       [3647, 4632, 'Manhattan', ..., 1, 365, 2],
       ...,
       [36485431, 23492952, 'Manhattan', ..., 1, 27, 2],
       [36485609, 30985759, 'Manhattan', ..., 6, 2, 1],
       [36487245, 68119814, 'Manhattan', ..., 1, 23, 1]], dtype=object)
```

In [52]:

```
airbnb.head()
```

Out[52]:

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_li
0	2539	2787	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	0.21	
1	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	0.38	
2	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	
3	3831	4869	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	4.64	
4	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	0.10	

In [53]:

```
airbnb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   host_id                         48895 non-null  int64
 2   neighbourhood_group             48895 non-null  object
 3   neighbourhood                   48895 non-null  object
 4   latitude                        48895 non-null  float64
 5   longitude                       48895 non-null  float64
 6   room_type                       48895 non-null  object
 7   price                           48895 non-null  int64
 8   minimum_nights                  48895 non-null  int64
 9   number_of_reviews               48895 non-null  int64
 10  reviews_per_month               38843 non-null  float64
 11  calculated_host_listings_count  48895 non-null  int64
 12  availability_365                48895 non-null  int64
 13  categorical_price               48895 non-null  category
dtypes: category(1), float64(3), int64(7), object(3)
memory usage: 4.9+ MB
```

[10 pts] Code complete data pipeline using sklearn mixins

In [54]:

```
from sklearn.model_selection import train_test_split
airbnb = pd.read_csv('datasets/airbnb/AB_NYC_2019.csv')
airbnb_true_vals = airbnb['price'].copy()
airbnb.drop(columns=["name", "host_name", "last_review","id", "host_id", "price"], axis=1, inplace=True)
cat_feat = ['neighbourhood_group', 'neighbourhood', 'room_type']
numeric_columns = airbnb.drop(columns=cat_feat,axis=1)
numeric_features = list(numeric_columns)
print(f'{numeric_features=}')

print(airbnb.info())
```

```
numeric_features=['latitude', 'longitude', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availa
bility_365']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   neighbourhood_group             48895 non-null  object
 1   neighbourhood                   48895 non-null  object
 2   latitude                        48895 non-null  float64
 3   longitude                       48895 non-null  float64
 4   room_type                       48895 non-null  object
 5   minimum_nights                  48895 non-null  int64
 6   number_of_reviews               48895 non-null  int64
```

```
 7   reviews_per_month               38843 non-null  float64
 8   calculated_host_listings_count  48895 non-null  int64
 9   availability_365                48895 non-null  int64
dtypes: float64(3), int64(4), object(3)
memory usage: 3.7+ MB
None
```

In [55]:

```
# imputer = SimpleImputer(strategy="median") # use median imputation for missing values
minimum_nights_ix = 2
number_of_reviews_ix = 3
reviews_per_month_ix = 4
availability_365_ix = 6

class AbnbAugmentFeatures(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X):
        v1 = X[:, number_of_reviews_ix]
        v2 = X[:, reviews_per_month_ix]
        num_months_posting = np.divide(v1,v2,out=np.zeros_like(v1),where= v2 != 0)
        v3 = X[:, minimum_nights_ix]
        v4 = X[:, availability_365_ix]
        one_stay_percent_of_year = np.divide(v3,v4,out=np.zeros_like(v3),where= v4 != 0)
        return np.c_[X, num_months_posting, one_stay_percent_of_year]

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('attribs_adder', AbnbAugmentFeatures()),
        ('std_scaler', StandardScaler()),
    ])

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, numeric_features),
        ("cat", OneHotEncoder(), cat_feat),
    ])

airbnb_prepared = full_pipeline.fit_transform(airbnb)
train_set, test_set, train_label, test_label = train_test_split(airbnb_prepared,
                                                                airbnb_true_vals,
                                                                test_size=0.25,
                                                                random_state=42)
```

[15 pts] Fit a model of your choice

The task is to predict the price, you could refer to the housing example on how to train and evaluate your model using Mean Absolute Error (MAE). Provide both test and train set MAE values.

In [56]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

lin_reg = LinearRegression()
lin_reg.fit(train_set, train_label)

train_pred = lin_reg.predict(train_set)
test_pred = lin_reg.predict(test_set)

train_mae = mean_absolute_error(train_label, train_pred)
test_mae = mean_absolute_error(test_label, test_pred)

print(f'{train_mae=}')
print(f'{test_mae=}')
```

```
train_mae=72.86928149499015
test_mae=68.79961790008474
```

2.4 3.0.4 Code complete data pipeline using sklearn mixing **10 / 10**

- ✓ **- 0 pts** **Correct**
- **- 5 pts** partially correct
- **- 10 pts** didn't do data pipeline

```
 7   reviews_per_month               38843 non-null  float64
 8   calculated_host_listings_count  48895 non-null  int64
 9   availability_365                48895 non-null  int64
dtypes: float64(3), int64(4), object(3)
memory usage: 3.7+ MB
None
```

In [55]:

```
# imputer = SimpleImputer(strategy="median") # use median imputation for missing values
minimum_nights_ix = 2
number_of_reviews_ix = 3
reviews_per_month_ix = 4
availability_365_ix = 6

class AbnbAugmentFeatures(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X):
        v1 = X[:, number_of_reviews_ix]
        v2 = X[:, reviews_per_month_ix]
        num_months_posting = np.divide(v1,v2,out=np.zeros_like(v1),where= v2 != 0)
        v3 = X[:, minimum_nights_ix]
        v4 = X[:, availability_365_ix]
        one_stay_percent_of_year = np.divide(v3,v4,out=np.zeros_like(v3),where= v4 != 0)
        return np.c_[X, num_months_posting, one_stay_percent_of_year]

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('attribs_adder', AbnbAugmentFeatures()),
        ('std_scaler', StandardScaler()),
    ])

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, numeric_features),
        ("cat", OneHotEncoder(), cat_feat),
    ])

airbnb_prepared = full_pipeline.fit_transform(airbnb)
train_set, test_set, train_label, test_label = train_test_split(airbnb_prepared,
                                                                airbnb_true_vals,
                                                                test_size=0.25,
                                                                random_state=42)
```

[15 pts] Fit a model of your choice

The task is to predict the price, you could refer to the housing example on how to train and evaluate your model using Mean Absolute Error (MAE). Provide both test and train set MAE values.

In [56]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

lin_reg = LinearRegression()
lin_reg.fit(train_set, train_label)

train_pred = lin_reg.predict(train_set)
test_pred = lin_reg.predict(test_set)

train_mae = mean_absolute_error(train_label, train_pred)
test_mae = mean_absolute_error(test_label, test_pred)

print(f'{train_mae=}')
print(f'{test_mae=}')
```

```
train_mae=72.86928149499015
test_mae=68.79961790008474
```

3 4. Fit a model of your choice **15 / 15**

✓ **- 0 pts** **Correct**

- 2.5 pts train MAE wrong or not provided. The MAE should be between 50-100ish for most cases (and not a very small number)

- 2.5 pts test MAE wrong or not provided. The MAE should be between 50-100ish for most cases (and not a very small number)

- 10 pts model fitting wrong