[32]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)

# let's try the full preprocessing pipeline on a few training instances
data = test_set.iloc[:5]
labels = housing_labels.iloc[:5]
data_prepared = full_pipeline.transform(data)

print("Predictions:", lin_reg.predict(data_prepared))
print("Actual labels:", list(labels))
```

```
Predictions: [425717.48517515 267643.98033218 227366.19892733 199614.48287493
 161425.25185885]
Actual labels: [286600.0, 340600.0, 196900.0, 46300.0, 254500.0]
```

We can evaluate our model using certain metrics, a fitting metric for regresison is the mean-squared-loss

$$L(\hat{Y}, Y) = \sum_{i}^{N} (\hat{y}_i - y_i)^2$$

where $\hat{y}$ is the predicted value, and y is the ground truth label.

[33]:
```
from sklearn.metrics import mean_squared_error

preds = lin_reg.predict(housing_prepared)
mse = mean_squared_error(housing_labels, preds)
rmse = np.sqrt(mse)
rmse
```

[33]:
```
67784.32202861732
```

1 TODO: Applying the end-end ML steps to a different dataset.

We will apply what we've learnt to another dataset (airbnb dataset). We will predict airbnb price based on other features.

2 [25 pts] Visualizing Data

2.0.1 [5 pts] Load the data + statistics

- load the dataset
- display the first few rows of the data
- drop the following columns: name, host_id, host_name, last_review
- display a summary of the statistics of the loaded data

- plot histograms for 3 features of your choice

[34]:
```
import sys
assert sys.version_info >= (3, 5)
import sklearn
assert sklearn.__version__ >= "0.20"

import numpy as np
import os
%matplotlib inline
import matplotlib.pyplot as plt

np.random.seed(42)

%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

ROOT_DIR = "."
IMAGES_PATH = os.path.join(ROOT_DIR, "images")
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_name, tight_layout=True, fig_extension="png", resolution=300):
    '''
        plt.savefig wrapper. refer to
        https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.savefig.html
    '''
    path = os.path.join(IMAGES_PATH, fig_name + "." + fig_extension)
    print("Saving figure", fig_name)
    if tight_layout:
            plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

[35]:
```
import os
import tarfile
import urllib
DATASET_PATH = os.path.join("datasets", "airbnb")
```

[36]:
```
import pandas as pd
def load_airbnb_data(airbnb_path):
    csv_path = os.path.join(airbnb_path, "AB_NYC_2019.csv")
    return pd.read_csv(csv_path)
```

[37]:
```
airbnb = load_airbnb_data(DATASET_PATH)
airbnb.head()
```

[37]:
```
     id                                             name  host_id  \
0  2539               Clean & quiet apt home by the park     2787
1  2595                            Skylit Midtown Castle     2845
2  3647              THE VILLAGE OF HARLEM…NEW YORK !     4632
3  3831                  Cozy Entire Floor of Brownstone     4869
4  5022  Entire Apt: Spacious Studio/Loft by central park     7192

     host_name neighbourhood_group neighbourhood  latitude  longitude  \
0         John            Brooklyn    Kensington  40.64749  -73.97237
1     Jennifer           Manhattan       Midtown  40.75362  -73.98377
2    Elisabeth           Manhattan        Harlem  40.80902  -73.94190
3  LisaRoxanne            Brooklyn  Clinton Hill  40.68514  -73.95976
4        Laura           Manhattan   East Harlem  40.79851  -73.94399

         room_type  price  minimum_nights  number_of_reviews last_review  \
0     Private room    149               1                  9  2018-10-19
1  Entire home/apt    225               1                 45  2019-05-21
2     Private room    150               3                  0         NaN
3  Entire home/apt     89               1                270  2019-07-05
4  Entire home/apt     80              10                  9  2018-11-19

   reviews_per_month  calculated_host_listings_count  availability_365
0               0.21                               6               365
1               0.38                               2               355
2                NaN                               1               365
3               4.64                               1               194
4               0.10                               1                 0
```

[38]:
```python
airbnb.drop(["id","name", "host_id", "host_name", "last_review"], axis=1,␣
↪inplace = True)
```

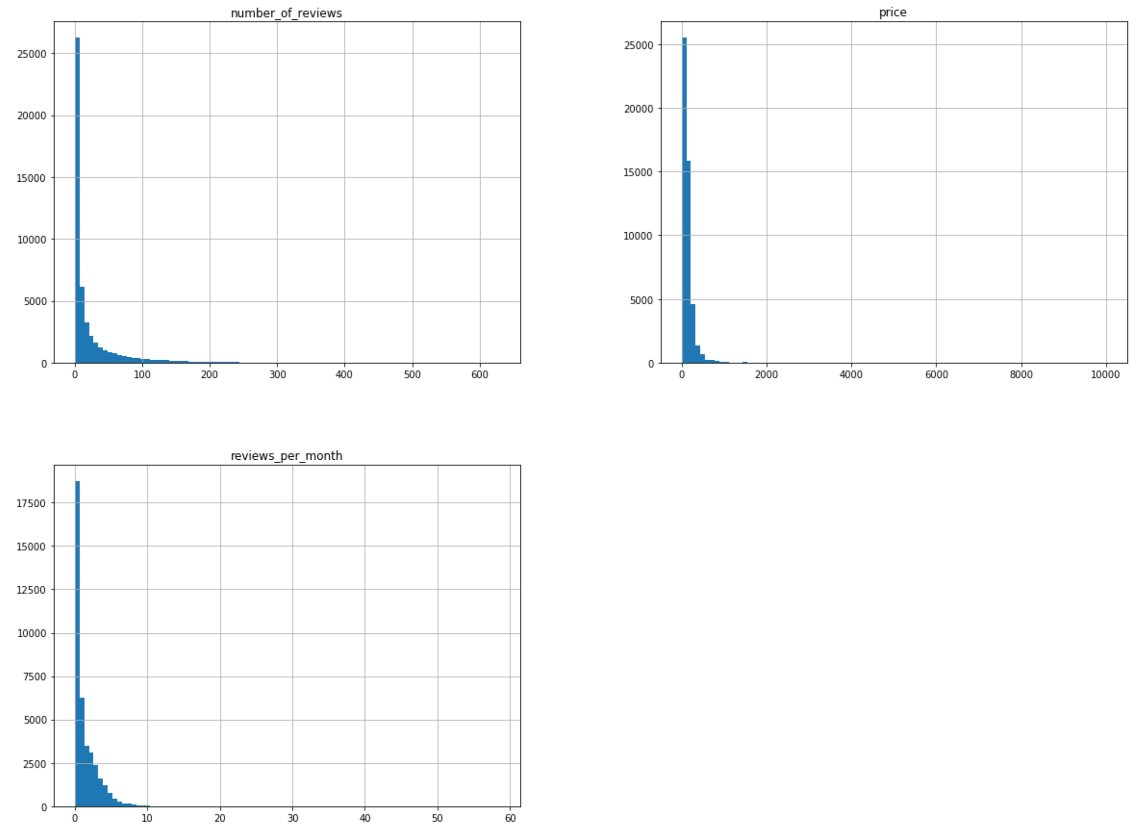
[39]:
```python
airbnb.describe()
```

[39]:
```
           latitude     longitude         price  minimum_nights  \
count  48895.000000  48895.000000  48895.000000    48895.000000
mean      40.728949    -73.952170    152.720687        7.029962
std        0.054530      0.046157    240.154170       20.510550
min       40.499790    -74.244420      0.000000        1.000000
25%       40.690100    -73.983070     69.000000        1.000000
50%       40.723070    -73.955680    106.000000        3.000000
75%       40.763115    -73.936275    175.000000        5.000000
max       40.913060    -73.712990  10000.000000     1250.000000

       number_of_reviews  reviews_per_month  calculated_host_listings_count  \
count       48895.000000       38843.000000                    48895.000000
mean           23.274466           1.373221                        7.143982
std            44.550582           1.680442                       32.952519
```

```
min              0.000000          0.010000                          1.000000
25%              1.000000          0.190000                          1.000000
50%              5.000000          0.720000                          1.000000
75%             24.000000          2.020000                          2.000000
max            629.000000         58.500000                        327.000000

       availability_365
count      48895.000000
mean         112.781327
std          131.622289
min            0.000000
25%            0.000000
50%           45.000000
75%          227.000000
max          365.000000
```

[40]:
```
airbnb.hist(["price", "number_of_reviews", "reviews_per_month"], bins =90,␣
 ↪figsize = (20,15)) #figsize just changes size of graph like a piece of paper
```

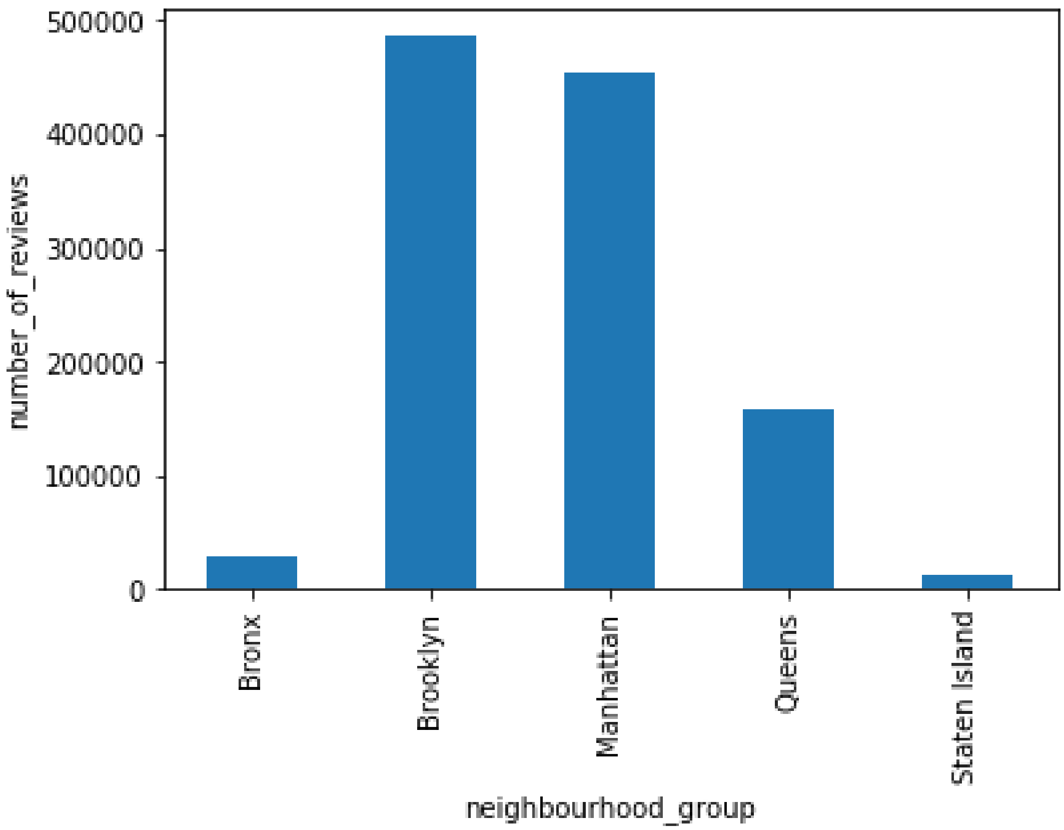
[40]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001AF1F886648>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001AF1F8B7708>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001AF1F8E6F08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001AF1F91EF48>]],
      dtype=object)
```



3 [5 pts] Plot total number_of_reviews per neighbourhood_group

[41]:
```
#airbnb["neighbourhood_group"].value_counts(["number_of_reviews"]).
 ↪plot(kind="bar")
airbnb.groupby("neighbourhood_group")["number_of_reviews"].agg("sum").
 ↪plot(kind="bar")
plt.ylabel("number_of_reviews")
```

[41]:
```
Text(0, 0.5, 'number_of_reviews')
```

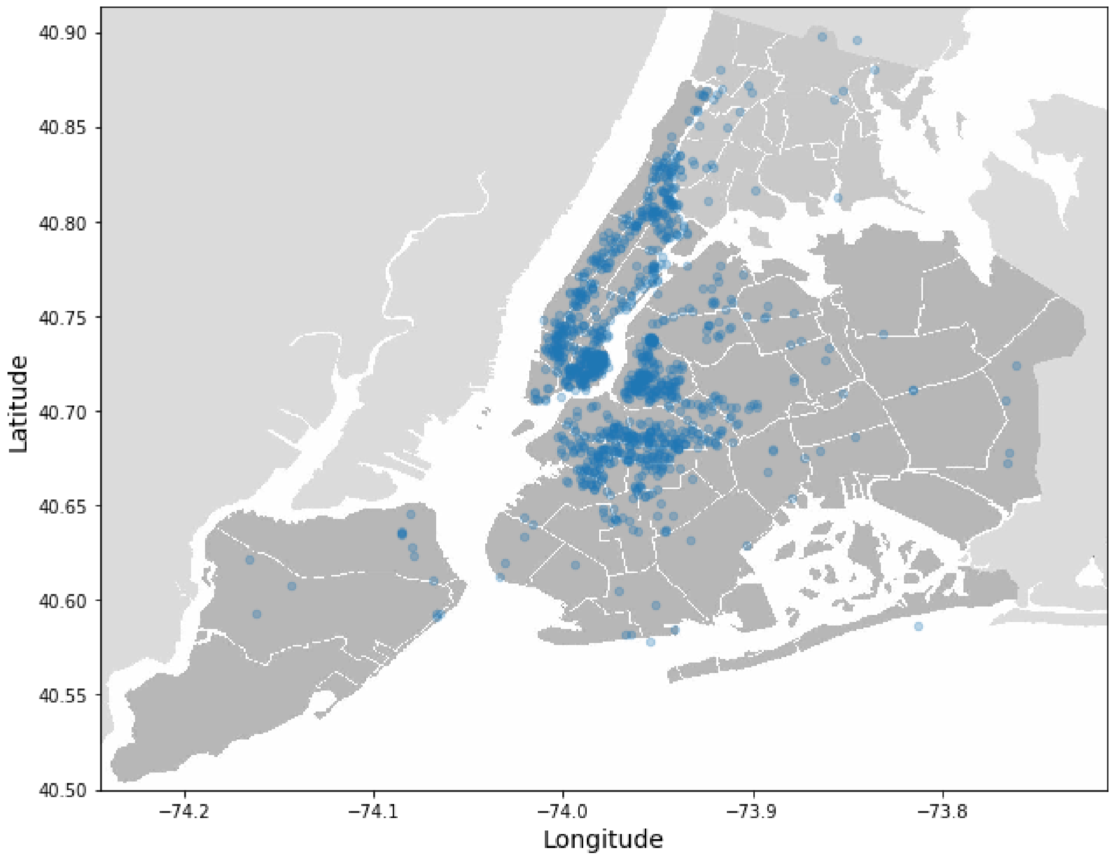


3.0.1 [5 pts] Plot map of airbnbs throughout New York (if it gets too crowded take a subset of the data, and try to make it look nice if you can :)).

[42]:
```
images_path = os.path.join('./', "images")
os.makedirs(images_path, exist_ok=True)
filename = "New_York.jpg"

import matplotlib.image as mpimg
NY_img=mpimg.imread(os.path.join(images_path, filename))
ax = airbnb.iloc[1:1239].plot(kind="scatter", x="longitude", y="latitude",␣
↪figsize=(10,7),
                       colorbar=False, alpha=0.3,
                      )
# overlay the new york map on the plotted scatter plot
# note: plt.imshow still refers to the most recent figure
# that hasn't been plotted yet.
plt.imshow(NY_img, extent=[-74.244420, -73.712990, 40.499790, 40.913060],␣
↪alpha=0.5,
           cmap=plt.get_cmap("jet"))
plt.ylabel("Latitude", fontsize=14)
```

```
plt.xlabel("Longitude", fontsize=14)

save_fig("NY_housing_prices_plot")
plt.show()
```

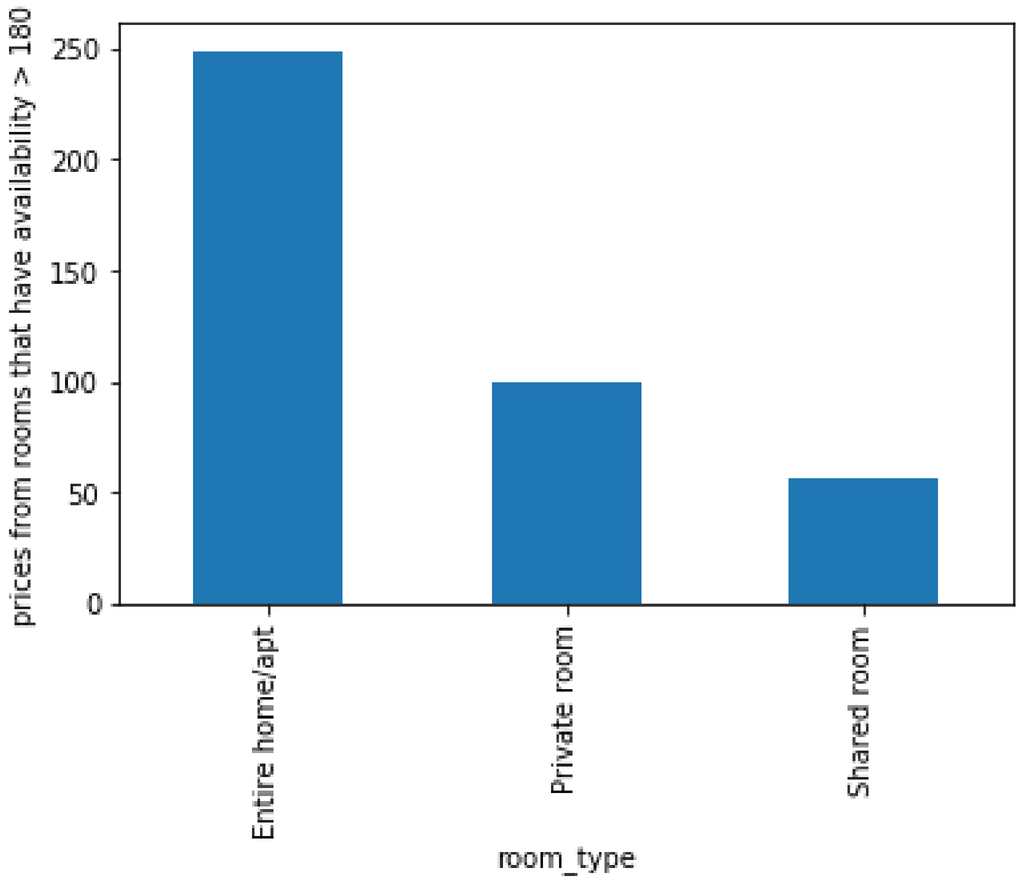
```
Saving figure NY_housing_prices_plot
```



3.0.2 [5 pts] Plot average price of room types who have availability greater than 180 days.

[43]:
```
#filter first then avg. loc filters
filter_prices = airbnb.loc[airbnb["availability_365"] > 180]
filter_prices.groupby("room_type").mean()["price"].plot(kind="bar")
plt.ylabel("prices from rooms that have availability > 180")
```

[43]:
```
Text(0, 0.5, 'prices from rooms that have availability > 180')
```

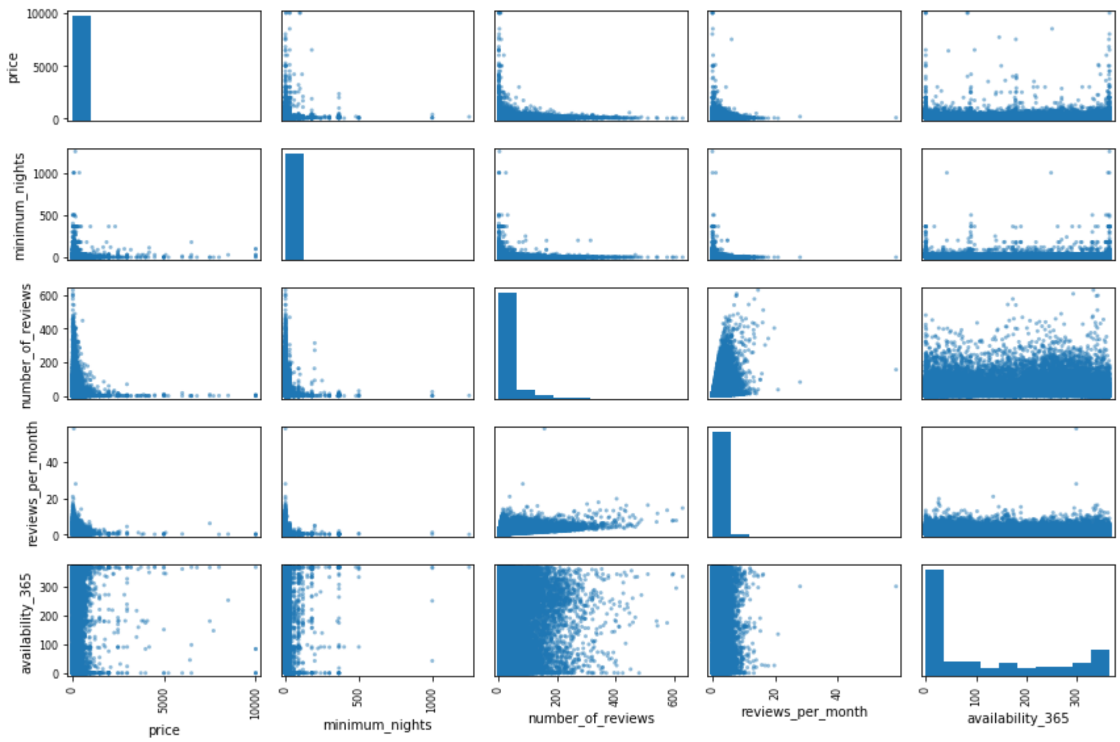


3.0.3 [5 pts] Plot correlation matrix

- which features have positive correlation?
- which features have negative correlation?

[44]:
```
from pandas.plotting import scatter_matrix
attributes =␣
↪["price","minimum_nights","number_of_reviews","reviews_per_month","availability_365"]
scatter_matrix(airbnb[attributes], figsize=(12,8))
save_fig("scatter_matrix_plot")
```

```
Saving figure scatter_matrix_plot
```



[45]:
```
airbnb.corr()
#Going to provide the correlations between only "price" (p), "minimum_nights"
↪(mn), "number_of_reviwes"(nr), "reviews_per_month"(rm),"availability_365"(a)
#Positive correlation between p and mn, p and a, mn and a, nr and rm, nr and a,
↪rm and a
#Negative correlation between p and nr, p and rm, mn and nr, mn and rm
```

[45]:
```
                                latitude  longitude     price  minimum_nights  \
latitude                        1.000000   0.084788  0.033939        0.024869
longitude                       0.084788   1.000000 -0.150019       -0.062747
price                           0.033939  -0.150019  1.000000        0.042799
minimum_nights                  0.024869  -0.062747  0.042799        1.000000
number_of_reviews              -0.015389   0.059094 -0.047954       -0.080116
reviews_per_month              -0.010142   0.145948 -0.030608       -0.121702
calculated_host_listings_count  0.019517  -0.114713  0.057472        0.127960
availability_365               -0.010983   0.082731  0.081829        0.144303

                                number_of_reviews  reviews_per_month  \
latitude                                -0.015389          -0.010142
longitude                                0.059094           0.145948
price                                   -0.047954          -0.030608
minimum_nights                          -0.080116          -0.121702
number_of_reviews                        1.000000           0.549868
```

```
reviews_per_month                        0.549868           1.000000
calculated_host_listings_count          -0.072376          -0.009421
availability_365                         0.172028           0.185791

                                calculated_host_listings_count  \
latitude                                              0.019517
longitude                                            -0.114713
price                                                 0.057472
minimum_nights                                        0.127960
number_of_reviews                                    -0.072376
reviews_per_month                                    -0.009421
calculated_host_listings_count                        1.000000
availability_365                                      0.225701

                                availability_365
latitude                               -0.010983
longitude                               0.082731
price                                   0.081829
minimum_nights                          0.144303
number_of_reviews                       0.172028
reviews_per_month                       0.185791
calculated_host_listings_count          0.225701
availability_365                        1.000000
```

4 [25 pts] Prepare the Data

4.0.1 [5 pts] Set aside 20% of the data as test test (80% train, 20% test).

[46]:

```python
from sklearn.model_selection import train_test_split
# let's first start by creating our train and test sets
airbnb_without_price = airbnb.drop("price", axis = 1)
X_train, X_test, y_train, y_test = train_test_split(airbnb_without_price, 
 ↪airbnb["price"], test_size = 0.2, random_state = 42)
print(airbnb_without_price)
```

```
      neighbourhood_group       neighbourhood  latitude  longitude  \
0                Brooklyn          Kensington  40.64749  -73.97237
1               Manhattan             Midtown  40.75362  -73.98377
2               Manhattan              Harlem  40.80902  -73.94190
3                Brooklyn        Clinton Hill  40.68514  -73.95976
4               Manhattan         East Harlem  40.79851  -73.94399
…                      …                   …        …         …
48890            Brooklyn  Bedford-Stuyvesant  40.67853  -73.94995
48891            Brooklyn            Bushwick  40.70184  -73.93317
48892           Manhattan              Harlem  40.81475  -73.94867
48893           Manhattan      Hell's Kitchen  40.75751  -73.99112
48894           Manhattan      Hell's Kitchen  40.76404  -73.98933
```

```
             room_type  minimum_nights  number_of_reviews  reviews_per_month  \
0         Private room               1                  9               0.21
1      Entire home/apt               1                 45               0.38
2         Private room               3                  0                NaN
3      Entire home/apt               1                270               4.64
4      Entire home/apt              10                  9               0.10
…                  …               …                …                 …
48890     Private room               2                  0                NaN
48891     Private room               4                  0                NaN
48892  Entire home/apt              10                  0                NaN
48893      Shared room               1                  0                NaN
48894     Private room               7                  0                NaN

       calculated_host_listings_count  availability_365
0                                   6               365
1                                   2               355
2                                   1               365
3                                   1               194
4                                   1                 0
…                                 …               …
48890                               2                 9
48891                               2                36
48892                               1                27
48893                               6                 2
48894                               1                23

[48895 rows x 10 columns]
```

4.0.2 [5 pts] Augment the dataframe with two other features which you think would be useful

[47]:
```python
airbnb["reviews_per_minimum_night"] = airbnb["number_of_reviews"]/
 ↪airbnb["minimum_nights"]
airbnb["availability_365_per_calculated_host_listings_count"] =␣
 ↪airbnb["availability_365"]/airbnb["calculated_host_listings_count"]

#print(airbnb["availability_365_per_calculated_host_listings_count"])
```

[48]:
```python
#new correlation matrix that shows the two new features got added into the␣
 ↪dataframe
airbnb.head()
```

[48]:
```
  neighbourhood_group neighbourhood  latitude  longitude        room_type  \
0            Brooklyn    Kensington  40.64749  -73.97237     Private room
1           Manhattan       Midtown  40.75362  -73.98377  Entire home/apt
2           Manhattan        Harlem  40.80902  -73.94190     Private room
```

```
3           Brooklyn  Clinton Hill  40.68514  -73.95976  Entire home/apt
4          Manhattan   East Harlem  40.79851  -73.94399  Entire home/apt

   price  minimum_nights  number_of_reviews  reviews_per_month  \
0    149               1                  9               0.21
1    225               1                 45               0.38
2    150               3                  0                NaN
3     89               1                270               4.64
4     80              10                  9               0.10

   calculated_host_listings_count  availability_365  \
0                               6               365
1                               2               355
2                               1               365
3                               1               194
4                               1                 0

   reviews_per_minimum_night  \
0                        9.0
1                       45.0
2                        0.0
3                      270.0
4                        0.9

   availability_365_per_calculated_host_listings_count
0                                          60.833333
1                                         177.500000
2                                         365.000000
3                                         194.000000
4                                           0.000000
```

4.0.3 [5 pts] Impute any missing feature with a method of your choice, and briefly discuss why you chose this imputation method

[60]:
```python
#like 1 line. Filling in all missing values
X_train["reviews_per_month"].fillna(0, inplace = True)
X_test["reviews_per_month"].fillna(0, inplace = True)
airbnb["reviews_per_month"].fillna(0, inplace = True)
#Decided to impute a feature that got rid of NaN values from the sets
#Imputed the features in the way I did because this was the only way I knew how␣
 ↪to
```

[50]:
```python
X_train.isna().sum()
#showing that X_train has no NaN values left.
```

[50]:
```
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
minimum_nights                    0
number_of_reviews                 0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

[51]:
```
X_test.isna().sum()
#showing that X_test has no NaN values left.
```

[51]:
```
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
minimum_nights                    0
number_of_reviews                 0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

[52]:
```
airbnb.isna().sum()
#showing that airbnb has no NaN values left.
```

[52]:
```
neighbourhood_group                                 0
neighbourhood                                       0
latitude                                            0
longitude                                           0
room_type                                           0
price                                               0
minimum_nights                                      0
number_of_reviews                                   0
reviews_per_month                                   0
calculated_host_listings_count                      0
availability_365                                    0
reviews_per_minimum_night                           0
availability_365_per_calculated_host_listings_count    0
dtype: int64
```

4.0.4 [10 pts] Code complete data pipeline using sklearn mixins

[53]:
```python
# This cell implements the complete pipeline for preparing the data
# using sklearns TransformerMixins
# Earlier we mentioned different types of features: categorical, and floats.
# In the case of floats we might want to convert them to categories.
# On the other hand categories in which are not already represented as integers␣
↪must be mapped to integers before
# feeding to the model.

# Additionally, categorical values could either be represented as one-hot␣
↪vectors or simple as normalized/unnormalized integers.
# Here we encode them using one hot vectors.

from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

from sklearn.base import BaseEstimator, TransformerMixin


imputer = SimpleImputer(strategy="constant", fill_value=0) # use median␣
↪imputation for missing values
# column index
X_train_num = X_train.drop(["neighbourhood_group", "room_type",␣
↪"neighbourhood"], axis=1)
min_nights_ix, num_reviews_ix, host_list_ix, availability_ix = 2, 3, 5, 6

#
class AugmentFeatures(BaseEstimator, TransformerMixin):
    '''
    implements the previous features we had defined
    airbnb["reviews_per_minimum_night"] = airbnb["number_of_reviews"]/
↪airbnb["minimum_nights"]
    airbnb["availability_365_per_calculated_host_listings_count"] =␣
↪airbnb["availability_365"]/airbnb["calculated_host_listings_count"]
    '''
    def __init__(self):
        pass
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X):
        reviews_per_minimum_night = X[:, num_reviews_ix] / X[:, min_nights_ix]
```

```
        availability_365_per_calculated_host_listings_count = X[:,␣
↪availability_ix] / X[:, host_list_ix]
        return np.c_[X, reviews_per_minimum_night,␣
↪availability_365_per_calculated_host_listings_count]

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="constant", fill_value = 0)),
        ('attribs_adder', AugmentFeatures()),
        ('std_scaler', StandardScaler()),
    ])

#housing_num_tr = num_pipeline.fit_transform(X_train)
numerical_features = list(X_train_num)
categorical_features = ["neighbourhood_group", "room_type", "neighbourhood"]

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, numerical_features),
        ("cat", OneHotEncoder(), categorical_features),
    ])

X_train_prepared = full_pipeline.fit_transform(X_train)
```

[54]:
```
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

from sklearn.base import BaseEstimator, TransformerMixin


imputer = SimpleImputer(strategy="constant", fill_value=0) # use median␣
↪imputation for missing values
# column index
X_test_num = X_test.drop(["neighbourhood_group", "room_type", "neighbourhood"],␣
↪axis=1)
min_nights_ix, num_reviews_ix, host_list_ix, availability_ix = 2, 3, 5, 6

#
class AugmentFeatures(BaseEstimator, TransformerMixin):
    '''
    implements the previous features we had defined
    airbnb["reviews_per_minimum_night"] = airbnb["number_of_reviews"]/
↪airbnb["minimum_nights"]
    airbnb["availability_365_per_calculated_host_listings_count"] =␣
↪airbnb["availability_365"]/airbnb["calculated_host_listings_count"]
```

```
    '''
    def __init__(self):
        pass
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X):
        reviews_per_minimum_night = X[:, num_reviews_ix] / X[:, min_nights_ix]
        availability_365_per_calculated_host_listings_count = X[:,␣
↪availability_ix] / X[:, host_list_ix]
        return np.c_[X, reviews_per_minimum_night,␣
↪availability_365_per_calculated_host_listings_count]

num_pipeline2 = Pipeline([
        ('imputer', SimpleImputer(strategy="constant", fill_value = 0)),
        ('attribs_adder', AugmentFeatures()),
        ('std_scaler', StandardScaler()),
    ])

#housing_num_tr = num_pipeline.fit_transform(X_train)
numerical_features = list(X_test_num)
categorical_features = ["neighbourhood_group", "room_type", "neighbourhood"]

full_pipeline2 = ColumnTransformer([
        ("num", num_pipeline2, numerical_features),
        ("cat", OneHotEncoder(), categorical_features),
    ])

X_test_prepared = full_pipeline2.fit_transform(X_test)
```

5 [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 MSE. Provide both test and train set MSE values.

[55]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train_prepared, y_train)

# let's try the full preprocessing pipeline on a few training instances
data = test_set.iloc[:5]
labels = y_train.iloc[:5]
data_prepared = full_pipeline.transform(X_train)

print("Predictions:", lin_reg.predict(data_prepared))
print("Actual labels:", list(labels))
```

```
Predictions: [180.86880796  46.06073467  71.52436815 … 271.19483893
244.40068087
 171.77465078]
Actual labels: [295, 70, 58, 75, 38]
```

[56]:
```python
from sklearn.metrics import mean_squared_error

preds = lin_reg.predict(X_train_prepared)
mse = mean_squared_error(y_train, preds)
print("mse=", mse)
```

```
mse= 53991.38474812648
```

[57]:
```python
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_test_prepared, y_test)

# let's try the full preprocessing pipeline on a few training instances
data = test_set.iloc[:5]
labels = y_test.iloc[:5]
data_prepared = full_pipeline2.transform(X_test)

print("Predictions:", lin_reg.predict(data_prepared))
print("Actual labels:", list(labels))
```

```
Predictions: [173.54630021  47.83012569 121.02791752 …  64.75298166
216.39221998
 182.42896259]
Actual labels: [89, 30, 120, 470, 199]
```

[58]:
```python
from sklearn.metrics import mean_squared_error

preds = lin_reg.predict(X_test_prepared)
mse = mean_squared_error(y_test, preds)
print("mse=", mse)
```

```
mse= 37309.3568533326
```