20W-COMSCI188 Project 1

STEWART DULANEY

TOTAL POINTS

65 / 65

QUESTION 1

1 Visualizing Data 25 / 25

✓ **- 0 pts Correct**

- 5 pts incorrect map of airbnbs throughout New York

- 5 pts incorrect map of airbnbs throughout New York

- 5 pts incorrect plot for total number_of_reviews per neighbourhood_group

- 5 pts incorrect plot for average price of room types who have availability greater than 180 days

- 3 pts missing some answers

QUESTION 2

2 Prepare the Data 25 / 25

✓ **- 0 pts Correct**

- 3 pts acceptable imputations are mean imputaiton (could be computed either globally or per id), median

- 5 pts Augment the dataframe with two other features which you thin.k would be useful

- 25 pts no answer

QUESTION 3

3 Fit a Model of Your Choice 15 / 15

✓ **- 0 pts Correct**

- 5 pts should Provide both test and train set MSE values.

- 15 pts no answer

gradescope

In [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
```

Out[33]:
```
67784.32202861732
```

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.

[25 pts] Visualizing Data

[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

In [34]:
```
def load_airbnb_data(airbnb_path):
    csv_path = os.path.join(airbnb_path, "AB_NYC_2019.csv")
    return pd.read_csv(csv_path)
AIRBNB_DATASET_PATH = os.path.join("datasets", "airbnb")
airbnb = load_airbnb_data(AIRBNB_DATASET_PATH)
```

In [35]:
```
airbnb.head()
```

Out[35]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	l
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-

In [36]:
```
airbnb_dropped = airbnb.copy()
airbnb_dropped = airbnb_dropped.drop(['name', 'host_id', 'host_name', 'last_review'], axis=1)
airbnb_dropped.head()
```

Out[36]:

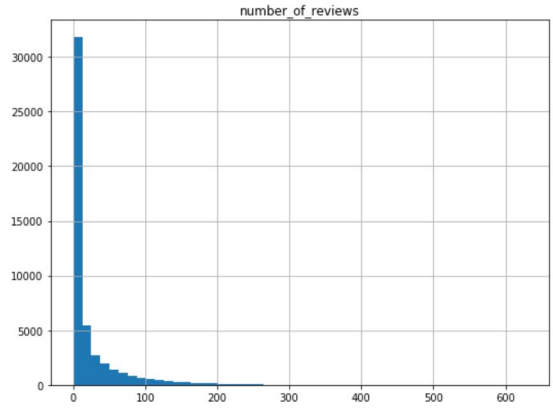
	id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_
0	2539	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	

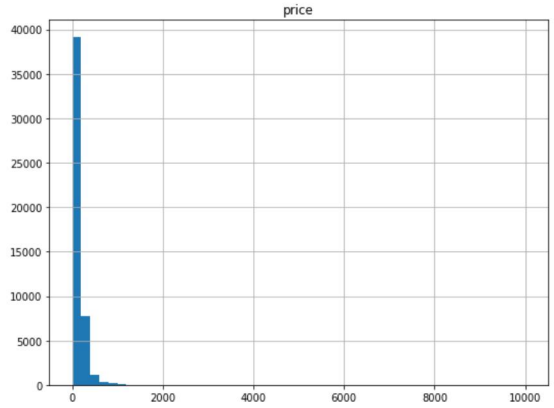
In [37]:
```
airbnb.describe()
```

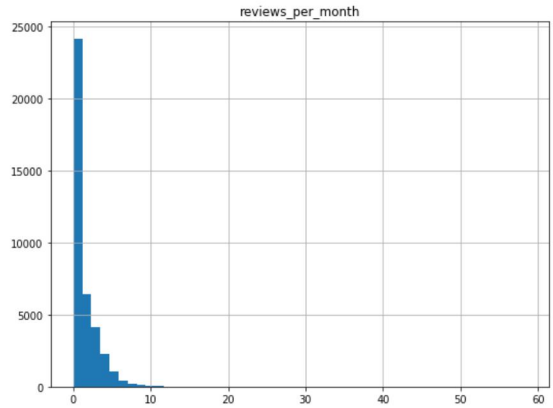
Out[37]:

	id	host_id	latitude	longitude	price	minimum_nights
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000

In [38]:
```
airbnb.hist(bins=50, figsize=(20,15), column=['price', 'number_of_reviews', 'reviews_per_month'])
plt.show()
```

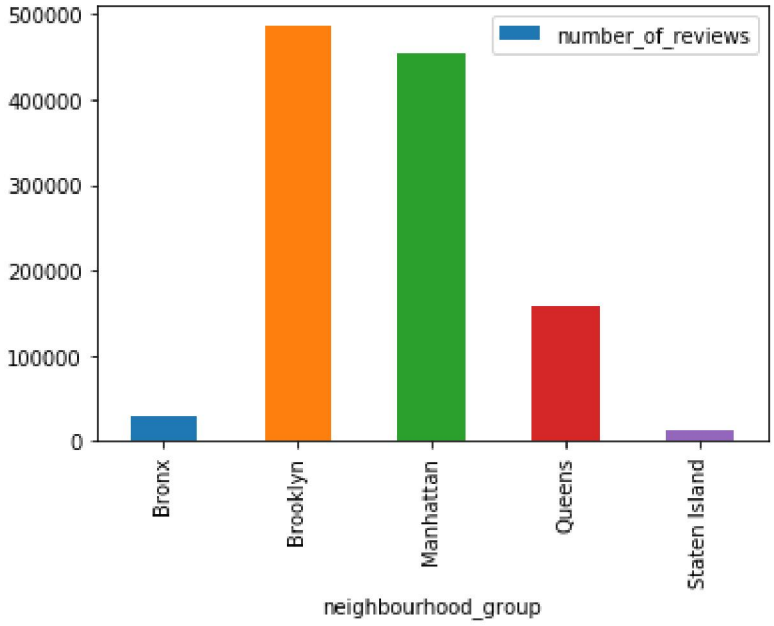






[5 pts] Plot total number_of_reviews per neighbourhood_group

In [39]:
```
airbnb.groupby('neighbourhood_group').number_of_reviews.sum().plot(legend=True, kind="bar")
plt.show()
```

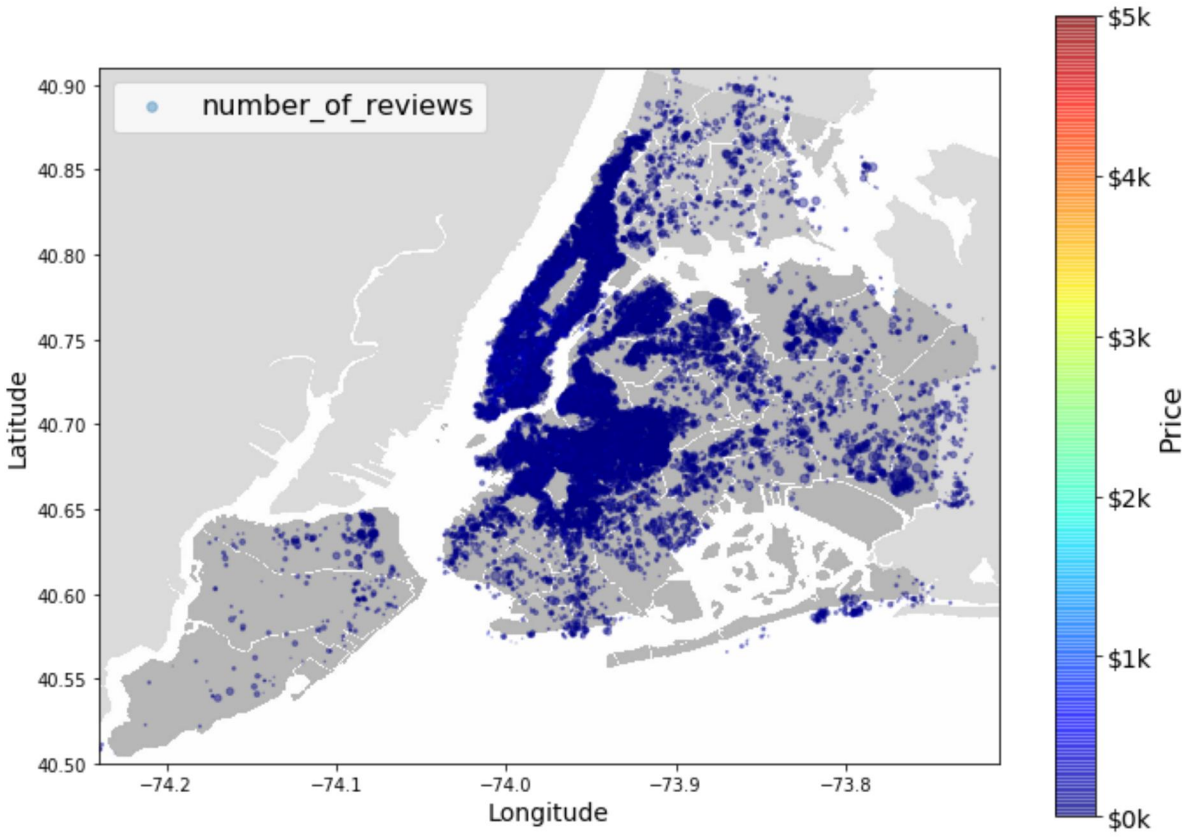


[5 pts] Plot map of airbnbs throughout New York (if it gets too crowded take a subset of the data, and try to make it look nice if you can :)).

In [40]:

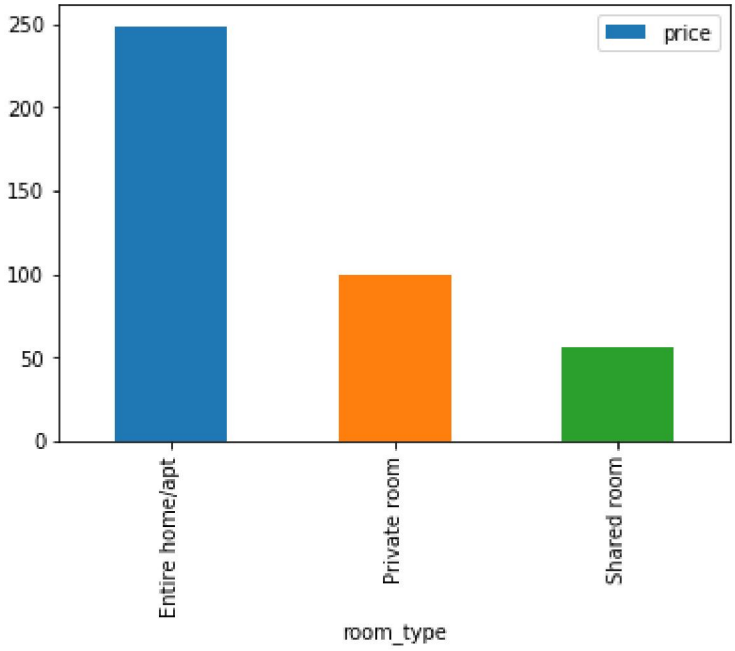
```
# Color code (heatmap) the dots based on price

# load an image of new york
images_path = os.path.join('./', "images")
os.makedirs(images_path, exist_ok=True)
filename = "newyork.png"

import matplotlib.image as mpimg
newyork_img=mpimg.imread(os.path.join(images_path, filename))
ax = airbnb.plot(kind="scatter", x="longitude", y="latitude", figsize=(1
0,7),
                         s=airbnb["number_of_reviews"]/10, label="number_o
f_reviews",
                         c="price", cmap=plt.get_cmap("jet"),
                         colorbar=False, alpha=0.4,
                        )
# 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(newyork_img, extent=[-74.24, -73.71, 40.50, 40.91]
           , alpha=0.5,
           cmap=plt.get_cmap("jet"))
plt.ylabel("Latitude", fontsize=14)
plt.xlabel("Longitude", fontsize=14)

# setting up heatmap colors based on price feature
prices = airbnb["price"]
tick_values = np.linspace(prices.min(), prices.max(), 11)
cb = plt.colorbar()
cb.ax.set_yticklabels(["$%dk"%(round(v/1000)) for v in tick_values], fon
tsize=14)
cb.set_label('Price', fontsize=16)

plt.legend(fontsize=16)
save_fig("newyork_housing_prices_plot")
plt.show()
```

```
Saving figure newyork_housing_prices_plot
```



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

In [41]:
```
airbnb_avail_gt_180 = airbnb.copy()
airbnb_avail_gt_180 = airbnb_avail_gt_180[airbnb_avail_gt_180["availabil
ity_365"] > 180]
airbnb_avail_gt_180.groupby("room_type").price.mean().plot(legend=True,
kind="bar")
plt.show()
```



[5 pts] Plot correlation matrix

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

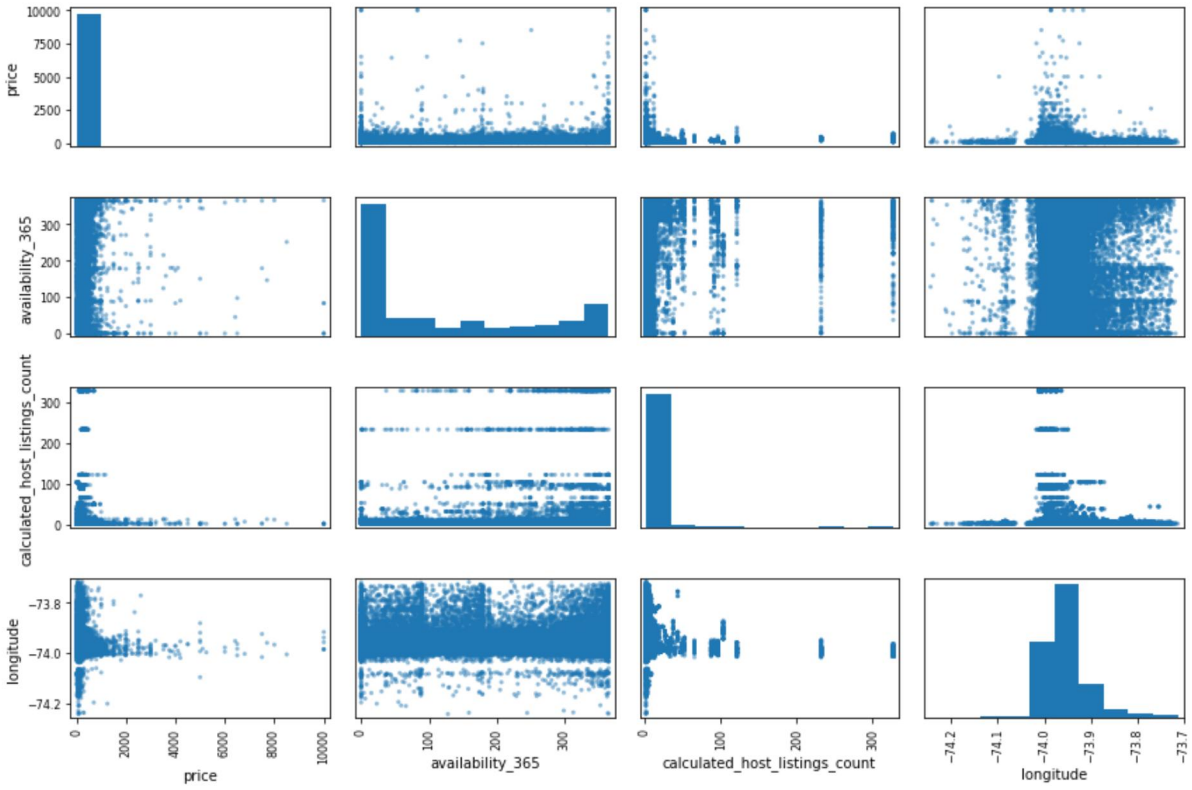
Although it was not obvious to me from the plots of the correlation matrices below, availability_365 and calculated_host_listings_count are the features with the greatest positive correlation to price, and longitude and number_of_reviews are the features with the greatest negative correlation. This is because all of the correlation values are close to 0, which essentially means no correlation.

In [42]:
```
# Determine which attributes to plot in correlation matrix
corr_matrix = airbnb.corr()
corr_matrix["price"].sort_values(ascending=False)
```

Out[42]:
```
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
```

In [43]:
```
from pandas.plotting import scatter_matrix
attributes = ["price", "availability_365", "calculated_host_listings_cou
nt", "longitude"]
scatter_matrix(airbnb[attributes], figsize=(12, 8))
save_fig("scatter_matrix_plot_airbnb")
```

```
Saving figure scatter_matrix_plot_airbnb
```



[25 pts] Prepare the Data

1 Visualizing Data **25 / 25**

✓ **- 0 pts** **Correct**

- 5 pts incorrect map of airbnbs throughout New York

- 5 pts incorrect map of airbnbs throughout New York

- 5 pts incorrect plot for total number_of_reviews per neighbourhood_group

- 5 pts incorrect plot for average price of room types who have availability greater than 180 days

- 3 pts missing some answers

gradescope

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

In [44]:
```
airbnb_augmented = airbnb.copy()
airbnb_augmented["price_per_host_listings_count"] = airbnb_augmented["pr
ice"]/airbnb_augmented["calculated_host_listings_count"]
airbnb_augmented["max_bookings"] = airbnb_augmented["availability_365"]/
airbnb_augmented["minimum_nights"]
airbnb_augmented.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 18 columns):
id                                48895 non-null int64
name                              48879 non-null object
host_id                           48895 non-null int64
host_name                         48874 non-null object
neighbourhood_group               48895 non-null object
neighbourhood                     48895 non-null object
latitude                          48895 non-null float64
longitude                         48895 non-null float64
room_type                         48895 non-null object
price                             48895 non-null int64
minimum_nights                    48895 non-null int64
number_of_reviews                 48895 non-null int64
last_review                       38843 non-null object
reviews_per_month                 38843 non-null float64
calculated_host_listings_count    48895 non-null int64
availability_365                  48895 non-null int64
price_per_host_listings_count     48895 non-null float64
max_bookings                      48895 non-null float64
dtypes: float64(5), int64(7), object(6)
memory usage: 6.7+ MB
```

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

As can be seen below, the only numerical feature with missing values is reviews_per_month. I chose to replace na values with median values because intuitively looking at the statistics from airbnb.describe(), it seems plausible reviews_per_month could have outliers on the upper end. For the next project I will figure out how to confirm this in code, but as a beginner that seemed out of scope for this project. Also, intuitively reviews_per_month seems like a good reason hosts could raise the price, so I didn't want to drop the complete feature. The categorical features with missing values are name, host_name, and last_review. I chose to drop the rows with missing values because I didn't know another technique to deal with this.

In [45]:
```
# Determine which columns have missing values
airbnb_imputed = airbnb_augmented.copy()
airbnb_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 18 columns):
id                                  48895 non-null int64
name                                48879 non-null object
host_id                             48895 non-null int64
host_name                           48874 non-null object
neighbourhood_group                 48895 non-null object
neighbourhood                       48895 non-null object
latitude                            48895 non-null float64
longitude                           48895 non-null float64
room_type                           48895 non-null object
price                               48895 non-null int64
minimum_nights                      48895 non-null int64
number_of_reviews                   48895 non-null int64
last_review                         38843 non-null object
reviews_per_month                   38843 non-null float64
calculated_host_listings_count      48895 non-null int64
availability_365                    48895 non-null int64
price_per_host_listings_count       48895 non-null float64
max_bookings                        48895 non-null float64
dtypes: float64(5), int64(7), object(6)
memory usage: 6.7+ MB
```

In [46]:
```
median = airbnb_imputed["reviews_per_month"].median()
airbnb_imputed["reviews_per_month"].fillna(median, inplace=True)
airbnb_imputed = airbnb_imputed.dropna(subset=["name", "host_name", "las
t_review"])
airbnb_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38821 entries, 0 to 48852
Data columns (total 18 columns):
id                                38821 non-null int64
name                              38821 non-null object
host_id                           38821 non-null int64
host_name                         38821 non-null object
neighbourhood_group               38821 non-null object
neighbourhood                     38821 non-null object
latitude                          38821 non-null float64
longitude                         38821 non-null float64
room_type                         38821 non-null object
price                             38821 non-null int64
minimum_nights                    38821 non-null int64
number_of_reviews                 38821 non-null int64
last_review                       38821 non-null object
reviews_per_month                 38821 non-null float64
calculated_host_listings_count    38821 non-null int64
availability_365                  38821 non-null int64
price_per_host_listings_count     38821 non-null float64
max_bookings                      38821 non-null float64
dtypes: float64(5), int64(7), object(6)
memory usage: 5.6+ MB
```

[10 pts] Code complete data pipeline using sklearn mixins

In [47]:

```
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

# use median imputation for missing values
imputer = SimpleImputer(strategy="median")
# remove the categorical and id features
airbnb_num = airbnb.copy()
airbnb_num = airbnb_num.drop(["id", "name", "host_id", "host_name", "nei
ghbourhood_group", "neighbourhood", "room_type", "last_review"], axis=1)
# column index
price_ix, calculated_host_listings_count_ix, availability_365_ix, minimu
m_nights_ix = 2, 6, 7, 3

airbnb = airbnb.dropna(subset=["name", "host_name", "last_review"])

class AugmentFeatures(BaseEstimator, TransformerMixin):
    '''
    implements the previous features we had defined
    '''
    def __init__(self):
        pass
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X):
        price_per_host_listings_count = X[:, price_ix] / X[:, calculated
_host_listings_count_ix]
        max_bookings = X[:, availability_365_ix] / X[:, minimum_nights_i
x]
        return np.c_[X, price_per_host_listings_count, max_bookings]

attr_adder = AugmentFeatures()

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

numerical_features = list(airbnb_num)
categorical_features = ["name", "host_name", "neighbourhood_group", "nei
ghbourhood", "room_type", "last_review"]

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, numerical_features),
        ("cat", OneHotEncoder(), categorical_features),
    ])
airbnb_prepared = full_pipeline.fit_transform(airbnb)
```

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

In [48]:
```
from sklearn.model_selection import train_test_split

airbnb_X = airbnb.drop("price", axis=1)
airbnb_y = airbnb["price"].copy()
X_train, X_test, y_train, y_test = train_test_split(airbnb_X, airbnb_y,
test_size=0.2, 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 MSE. Provide both test and train set MSE values.

In [49]:
```
# Test set

from sklearn.linear_model import LinearRegression

airbnb_labels = airbnb_y

lin_reg = LinearRegression()
lin_reg.fit(airbnb_prepared, airbnb_labels)

data = X_test
labels = y_test
data_prepared = full_pipeline.transform(data)

from sklearn.metrics import mean_squared_error

preds = lin_reg.predict(data_prepared)
mse = mean_squared_error(labels, preds)
mse
```

```
/Users/stewart/anaconda3/lib/python3.6/site-packages/pandas/core/indexi
ng.py:1472: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-lo
c-reindex-listlike
  return self._getitem_tuple(key)
```

Out[49]:
```
35169.35341014199
```

2 Prepare the Data **25 / 25**

✓ **- 0 pts Correct**

- 3 pts acceptable imputations are mean imputaiton (could be computed either globally or per id), median

- 5 pts Augment the dataframe with two other features which you thin.k would be useful

- 25 pts no answer

gradescope

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

In [48]:
```
from sklearn.model_selection import train_test_split

airbnb_X = airbnb.drop("price", axis=1)
airbnb_y = airbnb["price"].copy()
X_train, X_test, y_train, y_test = train_test_split(airbnb_X, airbnb_y,
test_size=0.2, 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 MSE. Provide both test and train set MSE values.

In [49]:
```
# Test set

from sklearn.linear_model import LinearRegression

airbnb_labels = airbnb_y

lin_reg = LinearRegression()
lin_reg.fit(airbnb_prepared, airbnb_labels)

data = X_test
labels = y_test
data_prepared = full_pipeline.transform(data)

from sklearn.metrics import mean_squared_error

preds = lin_reg.predict(data_prepared)
mse = mean_squared_error(labels, preds)
mse
```

```
/Users/stewart/anaconda3/lib/python3.6/site-packages/pandas/core/indexi
ng.py:1472: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-lo
c-reindex-listlike
  return self._getitem_tuple(key)
```

Out[49]:
```
35169.35341014199
```

In [50]:
```
# Train set

from sklearn.linear_model import LinearRegression

airbnb_labels = airbnb_y

lin_reg = LinearRegression()
lin_reg.fit(airbnb_prepared, airbnb_labels)

data = X_train
labels = y_train
data_prepared = full_pipeline.transform(data)

from sklearn.metrics import mean_squared_error

preds = lin_reg.predict(data_prepared)
mse = mean_squared_error(labels, preds)
mse
```

```
/Users/stewart/anaconda3/lib/python3.6/site-packages/pandas/core/indexi
ng.py:1472: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-lo
c-reindex-listlike
  return self._getitem_tuple(key)
```

Out[50]:
```
41846.880097970534
```

In [ ]:
```

```

3 Fit a Model of Your Choice **15 / 15**

✓ **- 0 pts** **Correct**

- 5 pts should Provide both test and train set MSE values.

- 15 pts no answer

gradescope