Data Imputation

Determine if there are any null-values and impute them.

```
sample_incomplete_rows = airbnb[airbnb.isnull().any(axis=1)].head()
sample_incomplete_rows
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190
19	7750	Huge 2 BR Upper East Cental Park	17985	Sing	Manhattan	East Harlem	40.79685	-73.94872
26	8700	Magnifique Suite au N de Manhattan - vue Cloitres	26394	Claude & Sophie	Manhattan	Inwood	40.86754	-73.92639
36	11452	Clean and Quiet in Brooklyn	7355	Vt	Brooklyn	Bedford-Stuyvesant	40.68876	-73.94312
38	11943	Country space in the city	45445	Harriet	Brooklyn	Flatbush	40.63702	-73.96327

```
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
airbnb['neighbourhood_group'] = labelencoder.fit_transform(airbnb['neighbourhood_group'])
airbnb['neighbourhood'] = labelencoder.fit_transform(airbnb['neighbourhood'])
airbnb['room_type'] = labelencoder.fit_transform(airbnb['room_type'])
airbnb = airbnb.dropna(subset=["reviews_per_month"])
airbnb = airbnb.dropna(subset=["price_cat"])

airbnb = airbnb.drop(['host_name', 'name'], axis=1)
airbnb.head()
#airbnb.dropna(subset=["price_cat"], inplace=True)
airbnb = airbnb.drop(['last_review'], axis=1)
airbnb.info()

airbnb.info()

# WRITE YOUR CODE HERE #
# sample_incomplete_rows.dropna(subset=["last_review"])    # option 1: simply drop rows that have null
# airbnb.info()
# sample_incomplete_rows = airbnb[airbnb.isnull().any(axis=1)].head()

# # sample_incomplete_rows.drop("total_bedrooms", axis=1)       # option 2: drop the complete feature

# # median = housing["total_bedrooms"].median()
# # sample_incomplete_rows["total_bedrooms"].fillna(median, inplace=True) # option 3: replace na value
# # sample_incomplete_rows

# sample_incomplete_rows = airbnb[airbnb.isnull().any(axis=1)].head()
# sample_incomplete_rows
# airbnb['neighbourhood_group'] = airbnb['neighbourhood_group'].astype('category')
# airbnb['room_type'] = airbnb['room_type'].astype('category')
# airbnb.drop(["name"], axis=1)
# airbnb['neighbourhood_group'] = airbnb['neighbourhood_group'].cat.codes
# airbnb['room_type'] = airbnb['room_type'].cat.codes
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 38833 entries, 0 to 48852
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              38833 non-null  int64
 1   host_id                         38833 non-null  int64
 2   neighbourhood_group             38833 non-null  int64
 3   neighbourhood                   38833 non-null  int64
 4   latitude                        38833 non-null  float64
 5   longitude                       38833 non-null  float64
 6   room_type                       38833 non-null  int64
 7   price                           38833 non-null  int64
 8   minimum_nights                  38833 non-null  int64
 9   number_of_reviews               38833 non-null  int64
 10  reviews_per_month               38833 non-null  float64
 11  calculated_host_listings_count  38833 non-null  int64
 12  availability_365                38833 non-null  int64
 13  price_cat                       38833 non-null  category
dtypes: category(1), float64(3), int64(10)
memory usage: 4.2 MB
<class 'pandas.core.frame.DataFrame'>
Index: 38833 entries, 0 to 48852
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              38833 non-null  int64
```

```
 1   host_id                         38833 non-null  int64
 2   neighbourhood_group             38833 non-null  int64
 3   neighbourhood                   38833 non-null  int64
 4   latitude                        38833 non-null  float64
 5   longitude                       38833 non-null  float64
 6   room_type                       38833 non-null  int64
 7   price                           38833 non-null  int64
 8   minimum_nights                  38833 non-null  int64
 9   number_of_reviews               38833 non-null  int64
 10  reviews_per_month               38833 non-null  float64
 11  calculated_host_listings_count  38833 non-null  int64
 12  availability_365                38833 non-null  int64
 13  price_cat                       38833 non-null  category
dtypes: category(1), float64(3), int64(10)
memory usage: 4.2 MB
```

Numeric Conversions

Finally, review what features in your dataset are non-numeric and convert them.

```
airbnb = airbnb.drop(['reviews_per_month'], axis=1)
airbnb.head()
```

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_
0	2539	2787	1	108	40.64749	-73.97237	1	149	
1	2595	2845	2	127	40.75362	-73.98377	0	225	
3	3831	4869	1	41	40.68514	-73.95976	0	89	
4	5022	7192	2	61	40.79851	-73.94399	0	80	
5	5099	7322	2	137	40.74767	-73.97500	0	200	

Prepare Data for Machine Learning

Using our `StratifiedShuffleSplit` function example from above, let's split our data into a 80/20 Training/Testing split using `price_cat` to partition the dataset

```
from sklearn.model_selection import StratifiedShuffleSplit
# let's first start by creating our train and test setspit

airbnb.reset_index(drop=True, inplace=True)

# Make sure there are no NaN values in 'price_cat'
# Handle NaN values here if any

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=0)

for train_index, test_index in split.split(airbnb, airbnb['price_cat']):
    strat_train_set = airbnb.loc[train_index]
    strat_test_set = airbnb.loc[test_index]
```

Finally, remove your labels `price` and `price_cat` from your testing and training cohorts, and create separate label features.

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



train_labels_price_cat = strat_train_set['price_cat'].copy()
test_labels_price_cat = strat_test_set['price_cat'].copy()

train_labels_price = strat_train_set['price'].copy()
test_labels_price = strat_test_set['price'].copy()

strat_train_set = strat_train_set.drop(['price_cat', 'price'], axis=1)
strat_test_set = strat_test_set.drop(['price_cat', 'price'], axis=1)
```

Fit a linear regression model

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**.

```
lin_reg_price_cat = LinearRegression()
lin_reg_price_cat.fit(strat_train_set, train_labels_price_cat)
```