```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
data = pd.read_csv("AB_NYC_2019.csv")
```

```
data.head()
```

latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review
40.64749	-73.97237	Private room	149	1	9	2018-10-19
40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21
40.80902	-73.94190	Private room	150	3	0	NaN
40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05
40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365                48895 non-null  int64
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
data['room_type'].value_counts()
```

```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

Lable Encoder

```
le = LabelEncoder()
data['room_type'] = le.fit_transform(data['room_type'])
```

```
data['room_type'].value_counts()
```

```
0    25409
1    22326
2     1160
Name: room_type, dtype: int64
```

```
le.classes_
```

```
array(['Entire home/apt', 'Private room', 'Shared room'], dtype=object)
```

One Hot Encoder

```
data['neighbourhood_group'].value_counts()
```

```
Manhattan        21661
Brooklyn         20104
Queens            5666
Bronx             1091
Staten Island      373
Name: neighbourhood_group, dtype: int64
```

```
oh = OneHotEncoder()
transformed_data = oh.fit_transform(data['neighbourhood_group'].values.reshape(-1,1)).toarray
```

```
oh.categories_
```

```
[array(['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island'],
       dtype=object)]
```

```
transformed_data = pd.DataFrame(transformed_data,columns=['Bronx', 'Brooklyn', 'Manhattan', '
```

```
transformed_data.head()
```

	Bronx	Brooklyn	Manhattan	Queens	Staten Island
0	0.0	1.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	1.0	0.0	0.0	0.0
4	0.0	0.0	1.0	0.0	0.0

```
transformed_data.iloc[100,]
```

```
Bronx            0.0
Brooklyn         0.0
Manhattan        1.0
Queens           0.0
Staten Island    0.0
Name: 100, dtype: float64
```

```
data['neighbourhood_group'][100]
```

```
'Manhattan'
```

Normalization and Standardization

```
numeric_col =[c for c in data.columns if data[c].dtype != np.dtype('O')]
```

```
len(numeric_col), len(data.columns)
```

```
(11, 16)
```

```
numeric_col
```

```
['id',
 'host_id',
 'latitude',
 'longitude',
 'room_type',
 'price',
       'minimum_nights',
       'number_of_reviews',
       'reviews_per_month',
       'calculated_host_listings_count',
       'availability_365']
```

```
temp = data[numeric_col]
```

```
temp
```

	id	host_id	latitude	longitude	room_type	price	minimum_nights	numbe
0	2539	2787	40.64749	-73.97237	1	149	1	
1	2595	2845	40.75362	-73.98377	0	225	1	
2	3647	4632	40.80902	-73.94190	1	150	3	
3	3831	4869	40.68514	-73.95976	0	89	1	
4	5022	7192	40.79851	-73.94399	0	80	10	
...	...	...	...	...	...	...	...	
48890	36484665	8232441	40.67853	-73.94995	1	70	2	
48891	36485057	6570630	40.70184	-73.93317	1	40	4	
48892	36485431	23492952	40.81475	-73.94867	0	115	10	
48893	36485609	30985759	40.75751	-73.99112	2	55	1	
48894	36487245	68119814	40.76404	-73.98933	1	90	7	

48895 rows × 11 columns

Normalization

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
normalizer = MinMaxScaler()
```

```
temp.dropna(axis = 1, inplace=True)
```

```
normalized_data = normalizer.fit_transform(temp)
```

```
pd.DataFrame(normalized_data,columns=temp.columns)
```

	id	host_id	latitude	longitude	room_type	price	minimum_nights	numbe
0	0.000000	0.000001	0.357393	0.511921	0.5	0.0149	0.000000	
1	0.000002	0.000001	0.614199	0.490469	0.0	0.0225	0.000000	
2	0.000030	0.000008	0.748252	0.569257	0.5	0.0150	0.001601	
3	0.000035	0.000009	0.448496	0.535649	0.0	0.0089	0.000000	
4	0.000068	0.000017	0.722820	0.565324	0.0	0.0080	0.007206	
...	...	...	...	...	...	...	...	
48890	0.999929	0.030002	0.432502	0.554109	0.5	0.0070	0.000801	
48891	0.999940	0.023944	0.488906	0.585684	0.5	0.0040	0.002402	
48892	0.999950	0.085632	0.762117	0.556517	0.0	0.0115	0.007206	
48893	0.999955	0.112946	0.623612	0.476639	1.0	0.0055	0.000000	
48894	1.000000	0.248315	0.639412	0.480007	0.5	0.0090	0.004804	

48895 rows × 10 columns

Standardization

```
stardard_scaler = StandardScaler()
```

```
s_data = stardard_scaler.fit_transform(temp)
```

```
pd.DataFrame(s_data,columns=temp.columns)
```

	id	host_id	latitude	longitude	room_type	price	minimum_nights	n
0	-1.731277	-0.860159	-1.493849	-0.437652	0.909359	-0.015493	-0.293996	
1	-1.731272	-0.860158	0.452436	-0.684639	-0.924247	0.300974	-0.293996	
2	-1.731176	-0.860135	1.468399	0.222497	0.909359	-0.011329	-0.196484	
3	-1.731159	-0.860132	-0.803398	-0.164450	-0.924247	-0.265335	-0.293996	

Handling with missing values

```
data.isnull().sum()
```

```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

Simple Imputer

```
from sklearn.impute import SimpleImputer
```

```
imputer = SimpleImputer(missing_values=np.nan , strategy='mean')
```

```
reviews_per_month_col = imputer.fit_transform(data['reviews_per_month'].values.reshape(-1,1)
```

```
pd.DataFrame(reviews_per_month_col).isnull().sum()
```

```
0    0
dtype: int64
```

```
data['reviews_per_month'].isnull().sum()
```

```
10052
```

Discretization

```
from sklearn.preprocessing import KBinsDiscretizer
```

```
temp.head()
```

ce	minimum_nights	number_of_reviews	calculated_host_listings_count	availability_365
49	1	9	6	365
25	1	45	2	355
50	3	0	1	365
89	1	270	1	194
80	10	9	1	0

Quantile Discretization Transform

```
trans = KBinsDiscretizer(n_bins=10,encode='ordinal',strategy='quantile')
new_data = trans.fit_transform(temp)
```

```
pd.DataFrame(new_data,columns=temp.columns)
```

	id	host id	latitude	longitude	room type	price	minimum nights	number of r

Double-click (or enter) to edit

Uniform Discretization Transform

```
trans = KBinsDiscretizer(n_bins=10, encode='ordinal',strategy='uniform')
new_data= trans.fit_transform(temp)
pd.DataFrame(new_data,columns=temp.columns)
```

	id	host_id	latitude	longitude	room_type	price	minimum_nights	number_of_r
0	0.0	0.0	3.0	5.0	5.0	0.0	0.0	
1	0.0	0.0	6.0	4.0	0.0	0.0	0.0	
2	0.0	0.0	7.0	5.0	5.0	0.0	0.0	
3	0.0	0.0	4.0	5.0	0.0	0.0	0.0	
4	0.0	0.0	7.0	5.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	
48890	9.0	0.0	4.0	5.0	5.0	0.0	0.0	
48891	9.0	0.0	4.0	5.0	5.0	0.0	0.0	
48892	9.0	0.0	7.0	5.0	0.0	0.0	0.0	
48893	9.0	1.0	6.0	4.0	9.0	0.0	0.0	
48894	9.0	2.0	6.0	4.0	5.0	0.0	0.0	

48895 rows × 10 columns

Quantile Discretization Transform

```
trans = KBinsDiscretizer(n_bins=10,encode='ordinal',strategy='kmeans')
new_data=trans.fit_transform(temp)
pd.DataFrame(new_data,columns=temp.columns)
```

```
        id   host_id  latitude  longitude  room_type  price  minimum_nights  number_of_r
0       0.0  0.0      1.0       4.0        1.0        0.0    0.0
1       0.0  0.0      6.0       4.0        0.0        1.0    0.0
2       0.0  0.0      8.0       5.0        1.0        0.0    1.0
3       0.0  0.0      3.0       5.0        0.0        0.0    0.0
4       0.0  0.0      7.0       5.0        0.0        0.0    2.0
...     ...  ...      ...       ...        ...        ...    ...
48890   9.0  0.0      2.0       5.0        1.0        0.0    1.0
48891   0.0  0.0      3.0       6.0        1.0        0.0    1.0

48893   9.0  1.0      6.0       3.0        2.0        0.0    0.0
48894   9.0  2.0      6.0       4.0        1.0        0.0    2.0
48895 rows × 10 columns
```

✓ 0s completed at 11:02 AM