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

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

```
data.head()
```

e	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviev
9	-73.97237	Private room	149	1	9	2018-10-19	
2	-73.98377	Entire home/apt	225	1	45	2019-05-21	
2	-73.94190	Private room	150	3	0	NaN	
4	-73.95976	Entire home/apt	89	1	270	2019-07-05	
1	-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
```

Encoding

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

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

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

Label Encoder

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

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

```
2    21661
1    20104
3     5666
0     1091
4      373
Name: neighbourhood_group, dtype: int64
```

```
le.classes_
```

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

OneHot Encoder

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

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

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

```
one_hot.categories_
```

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

```
transformed_data = pd.DataFrame(transformed_data ,
                                columns = ['Entire home/apt', 'Private room', 'Shared roon
```

```
transformed_data.head()
```

	Entire home/apt	Private room	Shared room
0	0.0	1.0	0.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0
3	1.0	0.0	0.0
4	1.0	0.0	0.0

```
transformed_data.iloc[90 , ]
```

```
Entire home/apt    0.0
Private room       1.0
Shared room        0.0
Name: 90, dtype: float64
```

```
data['room_type'][90]
```

```
'Private room'
```

Normalization & Standardization

```
# consider only numerical columns

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

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

```
(11, 16)
```

```
numeric_columns.remove('reviews_per_month')
```

```
temp_data = data[numeric_columns]
```

```
temp_data
```

```
                id   host_id  neighbourhood_group  price  minimum_nights  number_of_re
    0         2539      2787                    1    149               1
    1         2595      2845                    2    225               1
    2         3647      4632                    2    150               3
    3         3831      4869                    1     89               1
    4         5022      7192                    2     80              10
  ...          ...       ...                  ...    ...             ...
48890     36484665   8232441                    1     70               2
48891     36485057   6570630                    1     40               4
48892     36485431  23492952                    2    115              10
48893     36485609  30985759                    2     55               1
48894     36487245  68119814                    2     90               7

48895 rows × 8 columns
```

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

Normalization

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

```
normalizer = MinMaxScaler()
```

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

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

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

	id	host_id	neighbourhood_group	price	minimum_nights	number_of_rev
0	0.000000	0.000001	0.25	0.0149	0.000000	0.01
1	0.000002	0.000001	0.50	0.0225	0.000000	0.07
2	0.000030	0.000008	0.50	0.0150	0.001601	0.00
3	0.000035	0.000009	0.25	0.0089	0.000000	0.42
4	0.000068	0.000017	0.50	0.0080	0.007206	0.01
...	...	...	...	...	...	
48890	0.999929	0.030002	0.25	0.0070	0.000801	0.00
48891	0.999940	0.023944	0.25	0.0040	0.002402	0.00
48894	1.000000	0.248315	0.50	0.0090	0.004804	0.00

Standrization

```
standard_scaler = StandardScaler()

standardized_data = standard_scaler.fit_transform(temp_data)

pd.DataFrame(standardized_data , columns = temp_data.columns)
```

	id	host_id	neighbourhood_group	price	minimum_nights	number_of
0	-1.731277	-0.860159	-0.917828	-0.015493	-0.293996	
1	-1.731272	-0.860158	0.441222	0.300974	-0.293996	
2	-1.731176	-0.860135	0.441222	-0.011329	-0.196484	
3	-1.731159	-0.860132	-0.917828	-0.265335	-0.293996	
4	-1.731051	-0.860103	0.441222	-0.302811	0.144807	
...	...	...	...	...	...	
48890	1.590415	-0.755469	-0.917828	-0.344452	-0.245240	
48891	1.590451	-0.776609	-0.917828	-0.469373	-0.147729	
48892	1.590485	-0.561340	0.441222	-0.157070	0.144807	
48893	1.590501	-0.466024	0.441222	-0.406912	-0.293996	
48894	1.590650	0.006358	0.441222	-0.261171	-0.001461	

48895 rows × 8 columns

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
```

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

```
21
```

Simple Imputer

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

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

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

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

```
0    0
dtype: int64
```

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

```
10052
```

Discretization

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

```
temp_data.head()
```

	id	host_id	neighbourhood_group	price	minimum_nights	number_of_reviews	cal
0	2539	2787	1	149	1	9	
1	2595	2845	2	225	1	45	
2	3647	4632	2	150	3	0	
3	3831	4869	1	89	1	270	
4	5022	7192	2	80	10	9	

Quantile Discretization Transform

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

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

	id	host_id	neighbourhood_group	price	minimum_nights	number_of_reviews
0	0.0	0.0	1.0	6.0	0.0	4.0
1	0.0	0.0	2.0	8.0	0.0	6.0
2	0.0	0.0	2.0	6.0	2.0	0.0
3	0.0	0.0	1.0	3.0	0.0	7.0
4	0.0	0.0	2.0	3.0	4.0	4.0
...	...	...	...	...	...	...
48890	9.0	2.0	1.0	2.0	1.0	0.0
48891	9.0	2.0	1.0	0.0	3.0	0.0
48892	9.0	4.0	2.0	5.0	4.0	0.0
48893	9.0	5.0	2.0	1.0	0.0	0.0
48894	9.0	6.0	2.0	4.0	4.0	0.0

48895 rows × 8 columns

Uniform Discretization Transform

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

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

	id	host_id	neighbourhood_group	price	minimum_nights	number_of_reviews
0	0.0	0.0	2.0	0.0	0.0	0.0
1	0.0	0.0	5.0	0.0	0.0	0.0
2	0.0	0.0	5.0	0.0	0.0	0.0
3	0.0	0.0	2.0	0.0	0.0	4.0
4	0.0	0.0	5.0	0.0	0.0	0.0
...	...	...	...	...	...	...
48890	9.0	0.0	2.0	0.0	0.0	0.0
48891	9.0	0.0	2.0	0.0	0.0	0.0
48892	9.0	0.0	5.0	0.0	0.0	0.0
48893	9.0	1.0	5.0	0.0	0.0	0.0
48894	9.0	2.0	5.0	0.0	0.0	0.0

KMeans Discretization Transform

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

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

	id	host_id	neighbourhood_group	price	minimum_nights	number_of_reviews
0	0.0	0.0	1.0	0.0	0.0	0.0
1	0.0	0.0	2.0	1.0	0.0	2.0
2	0.0	0.0	2.0	0.0	1.0	0.0
3	0.0	0.0	1.0	0.0	0.0	7.0
4	0.0	0.0	2.0	0.0	2.0	0.0
...	...	...	...	...	...	...
48890	9.0	0.0	1.0	0.0	1.0	0.0
48891	9.0	0.0	1.0	0.0	1.0	0.0
48892	9.0	1.0	2.0	0.0	2.0	0.0
48893	9.0	1.0	2.0	0.0	0.0	0.0
48894	9.0	2.0	2.0	0.0	2.0	0.0

48895 rows × 8 columns

✓ 1s completed at 11:02 AM