```
!pip install category_encoders
```

```
Requirement already satisfied: category_encoders in c:\users\mario\
anaconda3\lib\site-packages (2.6.4)
Requirement already satisfied: numpy>=1.14.0 in c:\users\mario\
anaconda3\lib\site-packages (from category_encoders) (1.26.4)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\mario\
anaconda3\lib\site-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\mario\
anaconda3\lib\site-packages (from category_encoders) (1.11.4)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\mario\
anaconda3\lib\site-packages (from category_encoders) (0.14.0)
Requirement already satisfied: pandas>=1.0.5 in c:\users\mario\
anaconda3\lib\site-packages (from category_encoders) (2.1.4)
Requirement already satisfied: patsy>=0.5.1 in c:\users\mario\
anaconda3\lib\site-packages (from category_encoders) (0.5.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
mario\anaconda3\lib\site-packages (from pandas>=1.0.5-
>category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\mario\
anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders)
(2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\mario\
anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders)
(2023.3)
Requirement already satisfied: six in c:\users\mario\anaconda3\lib\
site-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\mario\
anaconda3\lib\site-packages (from scikit-learn>=0.20.0-
>category_encoders) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mario\
anaconda3\lib\site-packages (from scikit-learn>=0.20.0-
>category_encoders) (2.2.0)
Requirement already satisfied: packaging>=21.3 in c:\users\mario\
anaconda3\lib\site-packages (from statsmodels>=0.9.0-
>category_encoders) (23.1)
```

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\mario\
anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\mario\
anaconda3\lib\site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.3.2 in c:\users\mario\
anaconda3\lib\site-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in c:\users\mario\
anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mario\
anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

```
import numpy  as np
import pandas as pd
import category_encoders as ce

import matplotlib.pyplot as plt
%matplotlib inline

# Cargar el dataset

file_path = './Practica/airbnb-listings-extract.csv'
house_data = pd.read_csv(file_path, sep=';')

# Inspección básica
print(house_data.shape)
house_data.head(5).T                                    # visualizamos 5
primeras filas
```

```
(14780, 89)


0  \
ID
11210388
Listing Url
https://www.airbnb.com/rooms/11210388
Scrape ID
20170306202425
Last Scraped
2017-03-07
Name                                                The Loft-Full Bath-
Deck w/View
...
...
Cancellation Policy
moderate
Calculated host listings count
1.0
Reviews per Month
3.5
Geolocation                                          30.3373609355,-
97.8632766782
Features                          Host Is Superhost,Host Has Profile
Pic,Host Id...


1  \
ID
17471131
Listing Url
https://www.airbnb.com/rooms/17471131
```

```
Scrape ID
20170407214050
Last Scraped
2017-04-08
Name                                                    Claris I,
Friendly Rentals
...
...
Cancellation Policy
super_strict_30
Calculated host listings count
106.0
Reviews per Month
0.86
Geolocation
41.3896829422,2.17262543017
Features                          Host Has Profile Pic,Requires
License,Instant ...


2  \
ID
17584891
Listing Url
https://www.airbnb.com/rooms/17584891
Scrape ID
20170407214050
Last Scraped
2017-04-08
Name                                              Style Terrace Red,
Friendly Rentals
...
...
Cancellation Policy
super_strict_30
Calculated host listings count
106.0
Reviews per Month
NaN
Geolocation
41.3930345489,2.16217327868
Features                          Host Has Profile Pic,Requires
License,Instant ...


3  \
ID
5398030
Listing Url
```

```
https://www.airbnb.com/rooms/5398030
Scrape ID
20170407214050
Last Scraped
2017-04-08
Name                                                Picasso Suite 1.4
Paseo de Gracia
...
...
Cancellation Policy
strict
Calculated host listings count
24.0
Reviews per Month
1.09
Geolocation
41.3969668101,2.1674178103
Features                          Host Has Profile Pic,Host Identity
Verified,Re...


4
ID
18104606
Listing Url
https://www.airbnb.com/rooms/18104606
Scrape ID
20170407214050
Last Scraped
2017-04-08
Name                                                Smart City Centre
Apartment II
...
...
Cancellation Policy
flexible
Calculated host listings count
92.0
Reviews per Month
NaN
Geolocation
41.3886851936,2.15514963616
Features                          Host Has Profile Pic,Host Identity
Verified,Is...

[89 rows x 5 columns]
```

```
from sklearn.model_selection import train_test_split

full_df = pd.read_csv("./Practica/airbnb-listings-extract.csv",
```

```
sep=';', decimal='.')

train, test = train_test_split(full_df, test_size=0.2, shuffle=True,
random_state=0)

print(f'Dimensiones del dataset de training: {train.shape}')
print(f'Dimensiones del dataset de test: {test.shape}')

# Guardamos
train.to_csv('./Practica/airbnb_train.csv', sep=';', decimal='.')
test.to_csv('./Practica/airbnb_test.csv', sep=';', decimal='.')

# A partir de este momento cargamos el dataset de train y trabajamos
ÚNICAMENTE con él.

house_data = pd.read_csv('./Practica/airbnb_train.csv', sep=';',
decimal='.')

# Cargar dataset de prueba (test)
house_data_test = pd.read_csv('./Practica/airbnb_test.csv', sep=';',
decimal='.')

house_data.head(5).T
```

```
Dimensiones del dataset de training: (11824, 89)
Dimensiones del dataset de test: (2956, 89)


0  \
Unnamed: 0
2472
ID
5994463
Listing Url
https://www.airbnb.com/rooms/5994463
Scrape ID
20170407214119
Last Scraped
2017-04-08
...
...
Cancellation Policy
moderate
Calculated host listings count
2.0
Reviews per Month
0.5
Geolocation                                              40.4077318793,-
3.68481869733
Features                          Host Has Profile Pic,Is Location
```

```
Exact,Require...


1  \
Unnamed: 0
12299
ID
14136180
Listing Url
https://www.airbnb.com/rooms/14136180
Scrape ID
20170407214119
Last Scraped
2017-04-08
...
...
Cancellation Policy
flexible
Calculated host listings count
1.0
Reviews per Month
2.43
Geolocation                                                40.4158022422,-
3.70534037765
Features                           Host Has Profile Pic,Host Identity
Verified,Re...


2  \
Unnamed: 0
4024
ID
15520134
Listing Url
https://www.airbnb.com/rooms/15520134
Scrape ID
20170407214119
Last Scraped
2017-04-08
...
...
Cancellation Policy
moderate
Calculated host listings count
16.0
Reviews per Month
NaN
Geolocation                                                40.3890481626,-
3.74037392557
```

```
Features                            Host Has Profile Pic,Host Identity
Verified,Is...

                                                                     3
\
Unnamed: 0                                                       12692

ID                                                             8809721

Listing Url                         https://www.airbnb.com/rooms/8809721

Scrape ID                                                20170407214119

Last Scraped                                                2017-04-08

...                                                                ...

Cancellation Policy                                           strict

Calculated host listings count                                  97.0

Reviews per Month                                                NaN

Geolocation                                 40.4128140929,-3.70305247638

Features                            Host Has Profile Pic,Requires License



4
Unnamed: 0
11228
ID
1162707
Listing Url
https://www.airbnb.com/rooms/1162707
Scrape ID
20170407214119
Last Scraped
2017-04-08
...
...
Cancellation Policy
strict
Calculated host listings count
2.0
Reviews per Month
2.08
Geolocation                                          40.4386311984,-
3.71371613279
```

```
Features                          Host Has Profile Pic,Host Identity
Verified,Is...

[90 rows x 5 columns]
```

```
# Filtrar solo propiedades en Madrid
house_data = house_data[
    (house_data['Latitude'] >= 40.3) & (house_data['Latitude'] <=
40.5) &
    (house_data['Longitude'] >= -3.8) & (house_data['Longitude'] <= -
3.6)
]

house_data.describe()
```

```
         Unnamed: 0             ID     Scrape ID       Host ID  \
count  10535.000000  1.053500e+04  1.053500e+04  1.053500e+04
mean    7493.543427  1.040448e+07  2.017041e+13  3.739380e+07
std     4216.793670  5.517682e+06  4.328330e+00  3.452172e+07
min       40.000000  1.986400e+04  2.017041e+13  1.745300e+04
25%     3811.500000  5.807986e+06  2.017041e+13  7.821970e+06
50%     7538.000000  1.152272e+07  2.017041e+13  2.728103e+07
75%    11164.500000  1.535673e+07  2.017041e+13  5.704252e+07
max    14757.000000  1.810984e+07  2.017041e+13  1.247534e+08

       Host Response Rate  Host Listings Count  Host Total Listings
Count  \
count         9192.000000         10532.000000
10532.000000
mean            94.816253            10.050513
10.050513
std             15.284715            28.443693
28.443693
min              0.000000             0.000000
0.000000
25%            100.000000             1.000000
1.000000
50%            100.000000             2.000000
2.000000
75%            100.000000             5.000000
5.000000
max            100.000000           519.000000
519.000000

           Latitude     Longitude  Accommodates  ...  Number of
Reviews  \
count  10535.000000  10535.000000  10535.000000  ...
10535.000000
mean      40.420199     -3.698290      3.186047  ...
23.037969
```

```
std        0.019283      0.020853      1.985503  ...
38.212889
min       40.332908     -3.795734      1.000000  ...
0.000000
25%       40.410078     -3.707900      2.000000  ...
1.000000
50%       40.418339     -3.701711      2.000000  ...
7.000000
75%       40.427564     -3.694308      4.000000  ...
28.000000
max       40.499445     -3.600767     16.000000  ...
356.000000

       Review Scores Rating  Review Scores Accuracy  \
count           8216.000000             8200.000000
mean              91.558301                9.402927
std                9.123766                0.938708
min               20.000000                2.000000
25%               88.000000                9.000000
50%               94.000000               10.000000
75%               98.000000               10.000000
max              100.000000               10.000000

       Review Scores Cleanliness  Review Scores Checkin  \
count                8205.000000            8194.000000
mean                    9.318464               9.624725
std                     1.009876               0.791372
min                     2.000000               2.000000
25%                     9.000000               9.000000
50%                    10.000000              10.000000
75%                    10.000000              10.000000
max                    10.000000              10.000000

       Review Scores Communication  Review Scores Location  \
count                  8204.000000             8191.000000
mean                      9.647123                9.549017
std                       0.754314                0.758997
min                       2.000000                2.000000
25%                       9.000000                9.000000
50%                      10.000000               10.000000
75%                      10.000000               10.000000
max                      10.000000               10.000000

       Review Scores Value  Calculated host listings count  Reviews
per Month
count          8192.000000                    10535.000000
8323.000000
mean              9.205811                        7.768201
1.922428
std               0.966980                       19.933428
```

```
1.870367
min                2.000000                          1.000000
0.020000
25%                9.000000                          1.000000
0.480000
50%                9.000000                          2.000000
1.280000
75%               10.000000                          4.000000
2.880000
max               10.000000                        145.000000
13.080000

[8 rows x 37 columns]
```

```
# Configuración para mostrar todas las columnas y filas
pd.set_option('display.max_columns', None)  # Mostrar todas las
columnas
pd.set_option('display.max_rows', None)     # Mostrar todas las filas

house_data.isnull().any(), house_data.isnull().sum()
```

```
(Unnamed: 0                         False
 ID                                 False
 Listing Url                        False
 Scrape ID                          False
 Last Scraped                       False
 Name                               False
 Summary                             True
 Space                               True
 Description                         True
 Experiences Offered                False
 Neighborhood Overview               True
 Notes                               True
 Transit                             True
 Access                              True
 Interaction                         True
 House Rules                         True
 Thumbnail Url                       True
 Medium Url                          True
 Picture Url                         True
 XL Picture Url                      True
 Host ID                            False
 Host URL                           False
 Host Name                           True
 Host Since                          True
 Host Location                       True
 Host About                          True
 Host Response Time                  True
 Host Response Rate                  True
 Host Acceptance Rate                True
```

```
Host Thumbnail Url                 True
Host Picture Url                   True
Host Neighbourhood                 True
Host Listings Count                True
Host Total Listings Count          True
Host Verifications                 True
Street                            False
Neighbourhood                      True
Neighbourhood Cleansed            False
Neighbourhood Group Cleansed      False
City                               True
State                              True
Zipcode                            True
Market                             True
Smart Location                    False
Country Code                      False
Country                           False
Latitude                          False
Longitude                         False
Property Type                     False
Room Type                         False
Accommodates                      False
Bathrooms                          True
Bedrooms                           True
Beds                               True
Bed Type                          False
Amenities                          True
Square Feet                        True
Price                              True
Weekly Price                       True
Monthly Price                      True
Security Deposit                   True
Cleaning Fee                       True
Guests Included                   False
Extra People                      False
Minimum Nights                    False
Maximum Nights                    False
Calendar Updated                  False
Has Availability                   True
Availability 30                   False
Availability 60                   False
Availability 90                   False
Availability 365                  False
Calendar last Scraped             False
Number of Reviews                 False
First Review                       True
Last Review                        True
Review Scores Rating               True
Review Scores Accuracy             True
```

```
Review Scores Cleanliness           True
Review Scores Checkin               True
Review Scores Communication         True
Review Scores Location              True
Review Scores Value                 True
License                             True
Jurisdiction Names                  True
Cancellation Policy                False
Calculated host listings count     False
Reviews per Month                   True
Geolocation                        False
Features                           False
dtype: bool,
Unnamed: 0                             0
ID                                     0
Listing Url                            0
Scrape ID                              0
Last Scraped                           0
Name                                   0
Summary                              378
Space                               2846
Description                            6
Experiences Offered                    0
Neighborhood Overview               3956
Notes                               6515
Transit                             3998
Access                              4554
Interaction                         4554
House Rules                         3684
Thumbnail Url                       2014
Medium Url                          2014
Picture Url                           18
XL Picture Url                      2014
Host ID                                0
Host URL                               0
Host Name                              3
Host Since                             3
Host Location                         33
Host About                          3942
Host Response Time                  1343
Host Response Rate                  1343
Host Acceptance Rate               10535
Host Thumbnail Url                     3
Host Picture Url                       3
Host Neighbourhood                  2612
Host Listings Count                    3
Host Total Listings Count              3
Host Verifications                     7
Street                                 0
```

Neighbourhood	3561
Neighbourhood Cleansed	0
Neighbourhood Group Cleansed	0
City	4
State	39
Zipcode	349
Market	39
Smart Location	0
Country Code	0
Country	0
Latitude	0
Longitude	0
Property Type	0
Room Type	0
Accommodates	0
Bathrooms	38
Bedrooms	18
Beds	36
Bed Type	0
Amenities	128
Square Feet	10130
Price	8
Weekly Price	7853
Monthly Price	7875
Security Deposit	6001
Cleaning Fee	4274
Guests Included	0
Extra People	0
Minimum Nights	0
Maximum Nights	0
Calendar Updated	0
Has Availability	10535
Availability 30	0
Availability 60	0
Availability 90	0
Availability 365	0
Calendar last Scraped	0
Number of Reviews	0
First Review	2212
Last Review	2213
Review Scores Rating	2319
Review Scores Accuracy	2335
Review Scores Cleanliness	2330
Review Scores Checkin	2341
Review Scores Communication	2331
Review Scores Location	2344
Review Scores Value	2343
License	10338
Jurisdiction Names	10535

```
Cancellation Policy                    0
Calculated host listings count         0
Reviews per Month                   2212
Geolocation                            0
Features                               0
dtype: int64)
```

```
# Imputar con la moda para columnas categóricas
columns_with_mode = ['Room Type', 'Property Type', 'Cancellation
Policy']
for col in columns_with_mode:
    house_data[col] = house_data[col].fillna(house_data[col].mode()
[0])

# Imputar con la media para columnas numéricas
columns_with_mean = ['Price', 'Bathrooms', 'Bedrooms', 'Beds']
for col in columns_with_mean:
    house_data[col] = house_data[col].fillna(house_data[col].mean())

# Imputar con la mediana para otras columnas numéricas
columns_with_median = ['Minimum Nights', 'Maximum Nights',
'Availability 30']
for col in columns_with_median:
    house_data[col] = house_data[col].fillna(house_data[col].median())

house_data.dtypes
```

```
Unnamed: 0                          int64
ID                                  int64
Listing Url                        object
Scrape ID                           int64
Last Scraped                       object
Name                               object
Summary                            object
Space                              object
Description                        object
Experiences Offered                object
Neighborhood Overview              object
Notes                              object
Transit                            object
Access                             object
Interaction                        object
House Rules                        object
Thumbnail Url                      object
Medium Url                         object
Picture Url                        object
XL Picture Url                     object
Host ID                             int64
Host URL                           object
Host Name                          object
```

```
Host Since                          object
Host Location                       object
Host About                          object
Host Response Time                  object
Host Response Rate                 float64
Host Acceptance Rate                object
Host Thumbnail Url                  object
Host Picture Url                    object
Host Neighbourhood                  object
Host Listings Count                float64
Host Total Listings Count          float64
Host Verifications                  object
Street                              object
Neighbourhood                       object
Neighbourhood Cleansed              object
Neighbourhood Group Cleansed        object
City                                object
State                               object
Zipcode                             object
Market                              object
Smart Location                      object
Country Code                        object
Country                             object
Latitude                           float64
Longitude                          float64
Property Type                       object
Room Type                           object
Accommodates                         int64
Bathrooms                          float64
Bedrooms                           float64
Beds                               float64
Bed Type                            object
Amenities                           object
Square Feet                        float64
Price                              float64
Weekly Price                       float64
Monthly Price                      float64
Security Deposit                   float64
Cleaning Fee                       float64
Guests Included                      int64
Extra People                         int64
Minimum Nights                       int64
Maximum Nights                       int64
Calendar Updated                    object
Has Availability                    object
Availability 30                      int64
Availability 60                      int64
Availability 90                      int64
Availability 365                     int64
```

```
Calendar last Scraped              object
Number of Reviews                   int64
First Review                       object
Last Review                        object
Review Scores Rating              float64
Review Scores Accuracy            float64
Review Scores Cleanliness         float64
Review Scores Checkin             float64
Review Scores Communication       float64
Review Scores Location            float64
Review Scores Value               float64
License                            object
Jurisdiction Names                 object
Cancellation Policy                object
Calculated host listings count    float64
Reviews per Month                 float64
Geolocation                        object
Features                           object
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder

# Definir las columnas categóricas a codificar
columns_label_encode = ['Room Type', 'Property Type', 'Cancellation
Policy', 'Neighbourhood Group Cleansed']

# Inicializar el LabelEncoder
label_encoders = {}  # Guardar los encoders para cada columna

# Aplicar LabelEncoder a cada columna
for column in columns_label_encode:
    le = LabelEncoder()
    house_data[column] = le.fit_transform(house_data[column])
    label_encoders[column] = le  # Guardar el encoder para referencia
futura

# Ver los valores codificados y sus significados
for column, le in label_encoders.items():
    print(f"Categorías codificadas para '{column}':")
    for class_, encoded_value in zip(le.classes_,
le.transform(le.classes_)):
        print(f"  {class_}: {encoded_value}")
    print("\n")


house_data['Host Listings Count'].fillna(house_data['Host Listings
Count'].mode()[0], inplace=True)
```

```
Categorías codificadas para 'Room Type':
  Entire home/apt: 0
```

```
  Private room: 1
  Shared room: 2


Categorías codificadas para 'Property Type':
  Apartment: 0
  Bed & Breakfast: 1
  Boutique hotel: 2
  Bungalow: 3
  Camper/RV: 4
  Casa particular: 5
  Chalet: 6
  Condominium: 7
  Dorm: 8
  Earth House: 9
  Guest suite: 10
  Guesthouse: 11
  Hostel: 12
  House: 13
  Loft: 14
  Other: 15
  Serviced apartment: 16
  Tent: 17
  Townhouse: 18
  Villa: 19


Categorías codificadas para 'Cancellation Policy':
  flexible: 0
  moderate: 1
  strict: 2
  super_strict_30: 3
  super_strict_60: 4


Categorías codificadas para 'Neighbourhood Group Cleansed':
  Arganzuela: 0
  Barajas: 1
  Carabanchel: 2
  Centro: 3
  Chamartín: 4
  Chamberí: 5
  Ciudad Lineal: 6
  Fuencarral - El Pardo: 7
  Hortaleza: 8
  Latina: 9
  Moncloa - Aravaca: 10
  Moratalaz: 11
  Puente de Vallecas: 12
  Retiro: 13
```

```
  Salamanca: 14
  San Blas - Canillejas: 15
  Tetuán: 16
  Usera: 17
  Vicálvaro: 18
  Villa de Vallecas: 19
  Villaverde: 20
```

```
# Definir las columnas que queremos conservar
columns_to_keep = [
    'Host Listings Count', 'Latitude', 'Longitude', 'Accommodates',
'Guests Included',
    'Extra People', 'Minimum Nights', 'Maximum Nights', 'Availability
30',
    'Availability 60', 'Availability 90', 'Availability 365', 'Number
of Reviews',
    'Calculated host listings count', 'Bathrooms', 'Bedrooms', 'Beds',
'Price',  # Columna objetivo
    'Room Type', 'Property Type', 'Cancellation Policy',
'Neighbourhood Group Cleansed'
]

# Eliminar todas las columnas que no estén en la lista
house_data = house_data[columns_to_keep]

# Verificar el resultado
house_data.head(5).T
```

```
                                          0            1
2  \
Host Listings Count                    2.000000     1.000000   16.000000

Latitude                              40.407732    40.415802   40.389048

Longitude                             -3.684819    -3.705340   -3.740374

Accommodates                           4.000000     4.000000    1.000000

Guests Included                        1.000000     1.000000    1.000000

Extra People                           0.000000     0.000000    0.000000

Minimum Nights                         2.000000     1.000000    3.000000

Maximum Nights                      1125.000000  1125.000000   30.000000

Availability 30                        9.000000    15.000000   14.000000

Availability 60                       32.000000    39.000000   44.000000
```

Availability 90	52.000000	64.000000	74.000000
Availability 365	117.000000	208.000000	140.000000
Number of Reviews	12.000000	20.000000	0.000000
Calculated host listings count	2.000000	1.000000	16.000000
Bathrooms	1.000000	1.000000	1.500000
Bedrooms	1.000000	1.000000	1.000000
Beds	2.000000	2.000000	8.000000
Price	60.000000	50.000000	10.000000
Room Type	0.000000	0.000000	2.000000
Property Type	0.000000	0.000000	0.000000
Cancellation Policy	1.000000	0.000000	1.000000
Neighbourhood Group Cleansed	13.000000	3.000000	2.000000

	3	4
Host Listings Count	114.000000	2.000000
Latitude	40.412814	40.438631
Longitude	-3.703052	-3.713716
Accommodates	2.000000	2.000000
Guests Included	1.000000	1.000000
Extra People	0.000000	10.000000
Minimum Nights	10.000000	1.000000
Maximum Nights	1125.000000	1125.000000
Availability 30	0.000000	7.000000
Availability 60	6.000000	34.000000
Availability 90	36.000000	62.000000
Availability 365	311.000000	337.000000
Number of Reviews	0.000000	97.000000
Calculated host listings count	97.000000	2.000000
Bathrooms	3.000000	1.000000
Bedrooms	1.000000	1.000000
Beds	1.000000	1.000000
Price	30.000000	32.000000
Room Type	1.000000	1.000000
Property Type	0.000000	0.000000
Cancellation Policy	2.000000	2.000000
Neighbourhood Group Cleansed	3.000000	5.000000

```
# Configuración para mostrar todas las columnas y filas
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas
pd.set_option('display.max_rows', None)     # Mostrar todas las filas

house_data.isnull().any(), house_data.isnull().sum()
```

```
(Host Listings Count                False
 Latitude                           False
 Longitude                          False
 Accommodates                       False
 Guests Included                    False
 Extra People                       False
 Minimum Nights                     False
 Maximum Nights                     False
 Availability 30                    False
 Availability 60                    False
 Availability 90                    False
 Availability 365                   False
 Number of Reviews                  False
 Calculated host listings count     False
 Bathrooms                          False
 Bedrooms                           False
 Beds                               False
 Price                              False
 Room Type                          False
 Property Type                      False
 Cancellation Policy                False
 Neighbourhood Group Cleansed       False
 dtype: bool,
 Host Listings Count                0
 Latitude                           0
 Longitude                          0
 Accommodates                       0
 Guests Included                    0
 Extra People                       0
 Minimum Nights                     0
 Maximum Nights                     0
 Availability 30                    0
 Availability 60                    0
 Availability 90                    0
 Availability 365                   0
 Number of Reviews                  0
 Calculated host listings count     0
 Bathrooms                          0
 Bedrooms                           0
 Beds                               0
 Price                              0
 Room Type                          0
 Property Type                      0
```

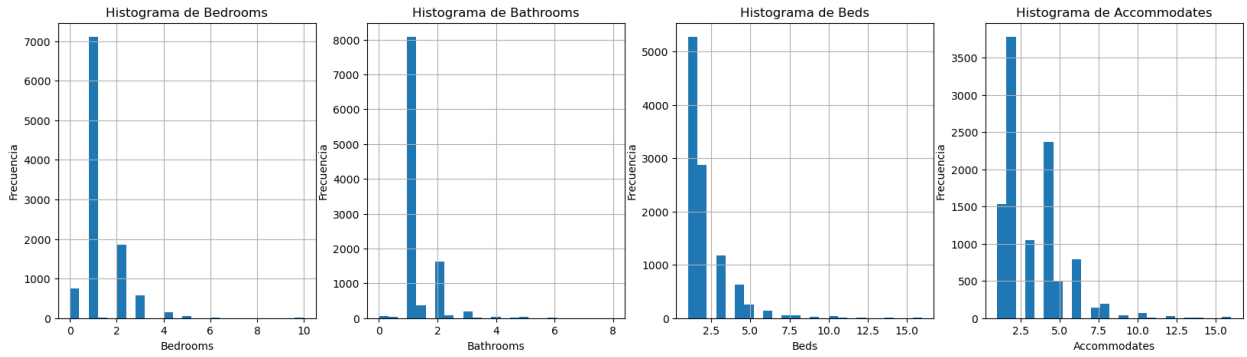
```
 Cancellation Policy               0
 Neighbourhood Group Cleansed      0
 dtype: int64)
```

```
house_data.dtypes
```

```
Host Listings Count               float64
Latitude                          float64
Longitude                         float64
Accommodates                        int64
Guests Included                     int64
Extra People                        int64
Minimum Nights                      int64
Maximum Nights                      int64
Availability 30                     int64
Availability 60                     int64
Availability 90                     int64
Availability 365                    int64
Number of Reviews                   int64
Calculated host listings count    float64
Bathrooms                         float64
Bedrooms                          float64
Beds                              float64
Price                             float64
Room Type                           int32
Property Type                       int32
Cancellation Policy                 int32
Neighbourhood Group Cleansed        int32
dtype: object
```

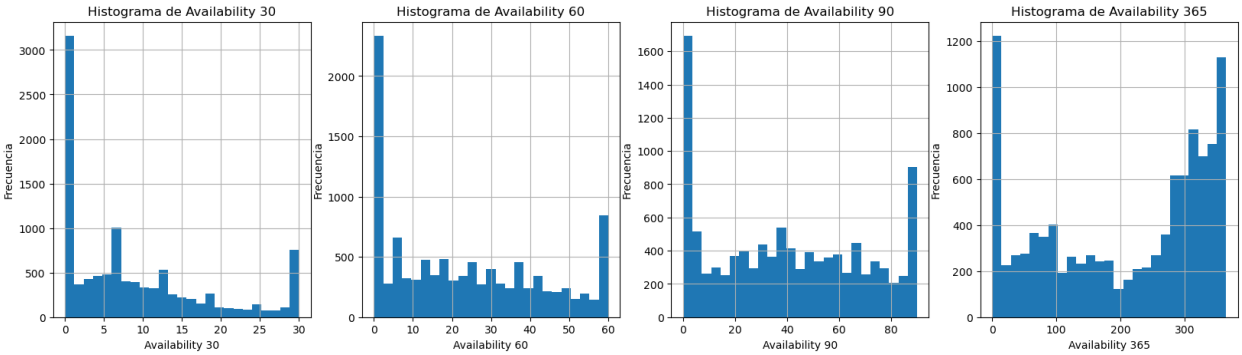
```
# Histogramas para las columnas seleccionadas

plt.figure(figsize=(20, 5))

plt.subplot(1,4,1)
house_data['Bedrooms'].plot.hist(alpha=1, bins=25, grid = True)
plt.title(f'Histograma de Bedrooms')
plt.xlabel('Bedrooms')
plt.ylabel('Frecuencia')

plt.subplot(1,4,2)
house_data['Bathrooms'].plot.hist(alpha=1, bins=25, grid = True)
plt.title(f'Histograma de Bathrooms')
plt.xlabel('Bathrooms')
plt.ylabel('Frecuencia')

plt.subplot(1,4,3)
house_data['Beds'].plot.hist(alpha=1, bins=25, grid = True)
plt.title(f'Histograma de Beds')
plt.xlabel('Beds')
```

```
plt.ylabel('Frecuencia')

plt.subplot(1,4,4)
house_data['Accommodates'].plot.hist(alpha=1, bins=25, grid = True)
plt.title(f'Histograma de Accommodates')
plt.xlabel('Accommodates')
plt.ylabel('Frecuencia')

plt.show()
```

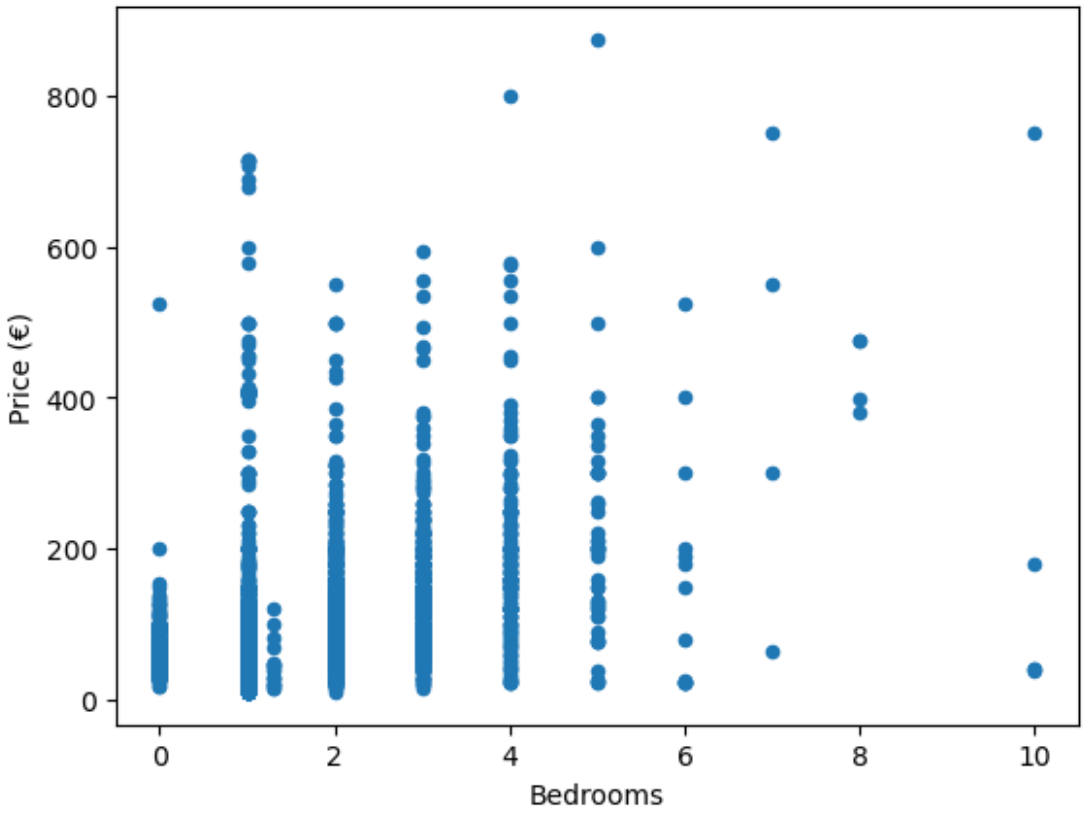


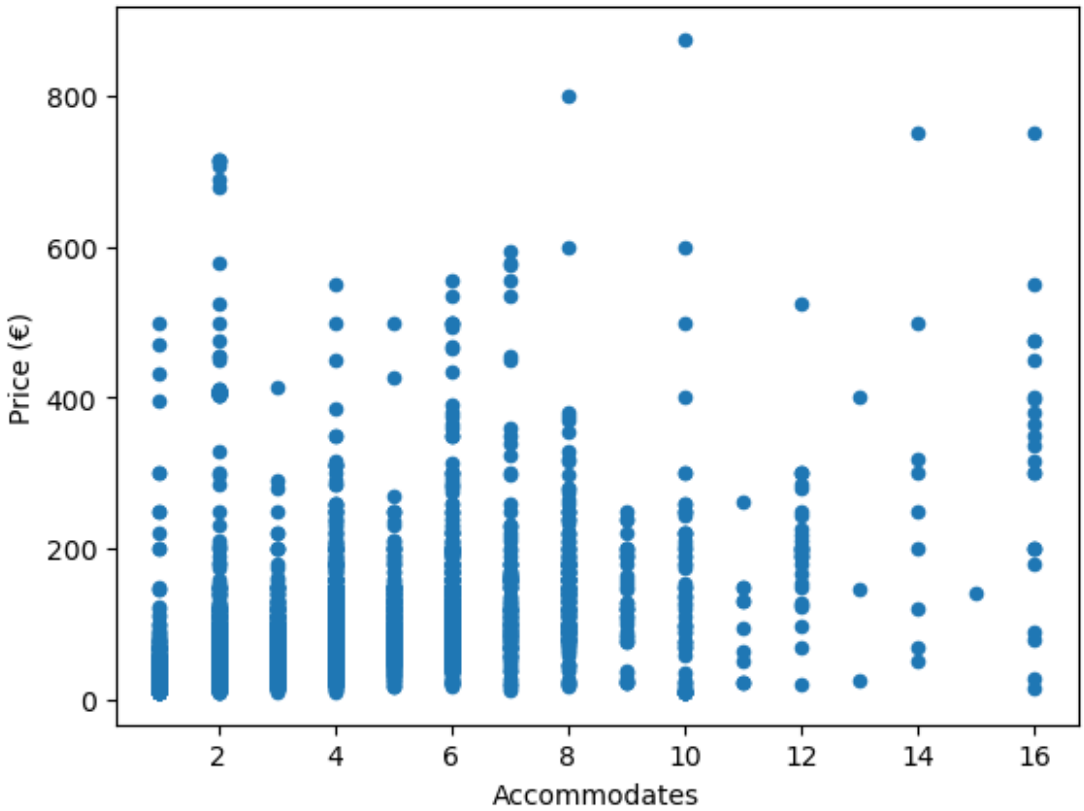

```
# Histogramas para las columnas seleccionadas

plt.figure(figsize=(20, 5))

plt.subplot(1,4,1)
house_data['Availability 30'].plot.hist(alpha=1, bins=25, grid = True)
plt.title(f'Histograma de Availability 30')
plt.xlabel('Availability 30')
plt.ylabel('Frecuencia')

plt.subplot(1,4,2)
house_data['Availability 60'].plot.hist(alpha=1, bins=25, grid = True)
plt.title(f'Histograma de Availability 60')
plt.xlabel('Availability 60')
plt.ylabel('Frecuencia')

plt.subplot(1,4,3)
house_data['Availability 90'].plot.hist(alpha=1, bins=25, grid = True)
plt.title(f'Histograma de Availability 90')
plt.xlabel('Availability 90')
plt.ylabel('Frecuencia')

plt.subplot(1,4,4)
house_data['Availability 365'].plot.hist(alpha=1, bins=25, grid =
True)
plt.title(f'Histograma de Availability 365')
plt.xlabel('Availability 365')
plt.ylabel('Frecuencia')
```

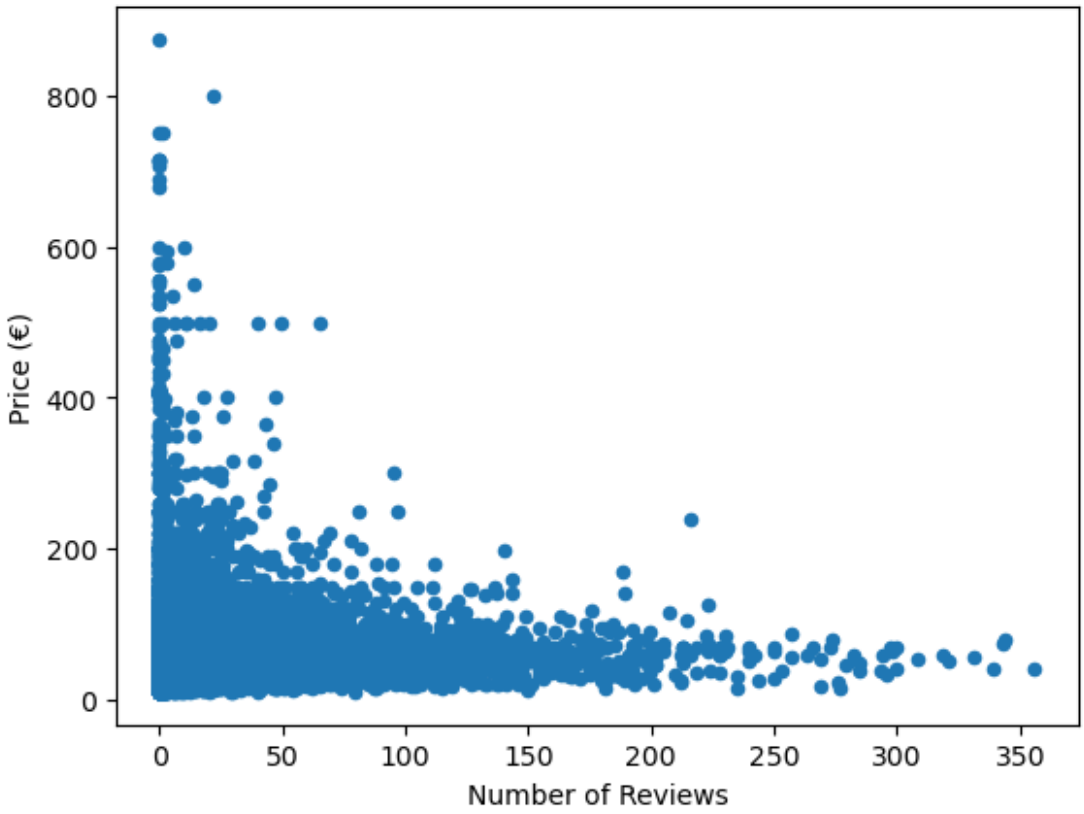
```
plt.show()
```

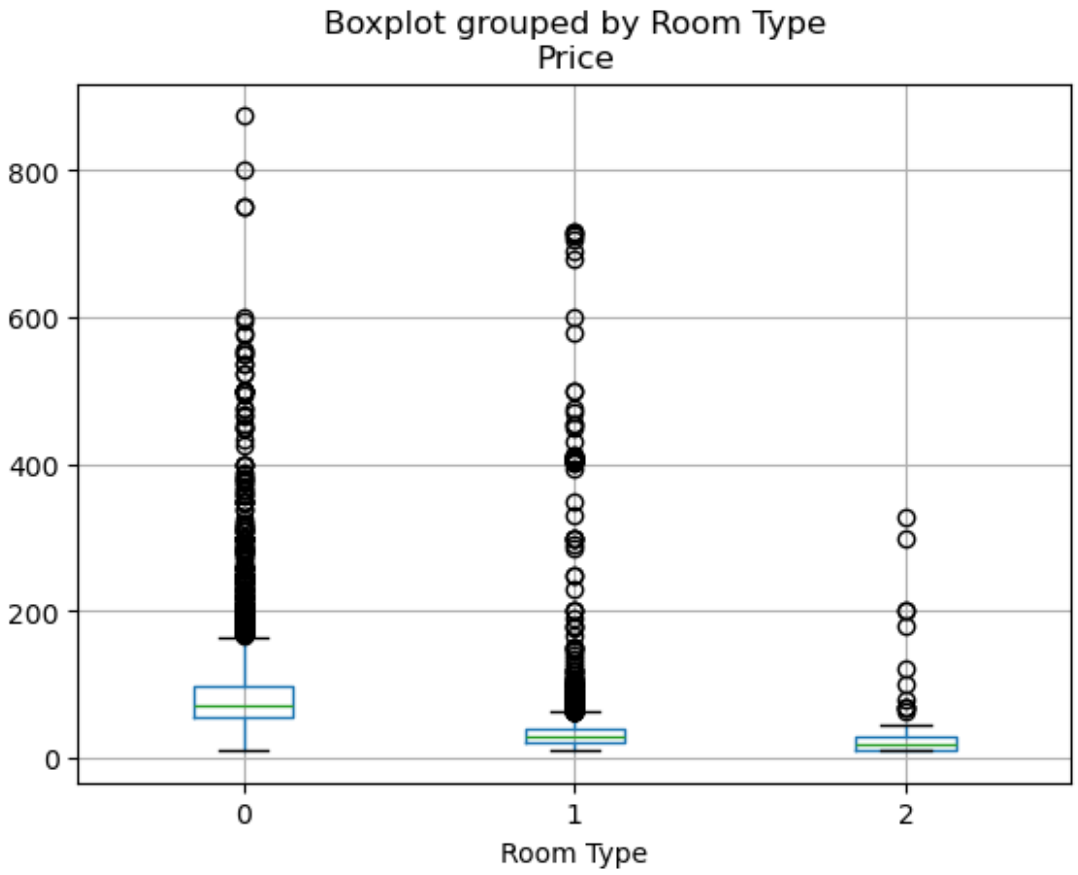


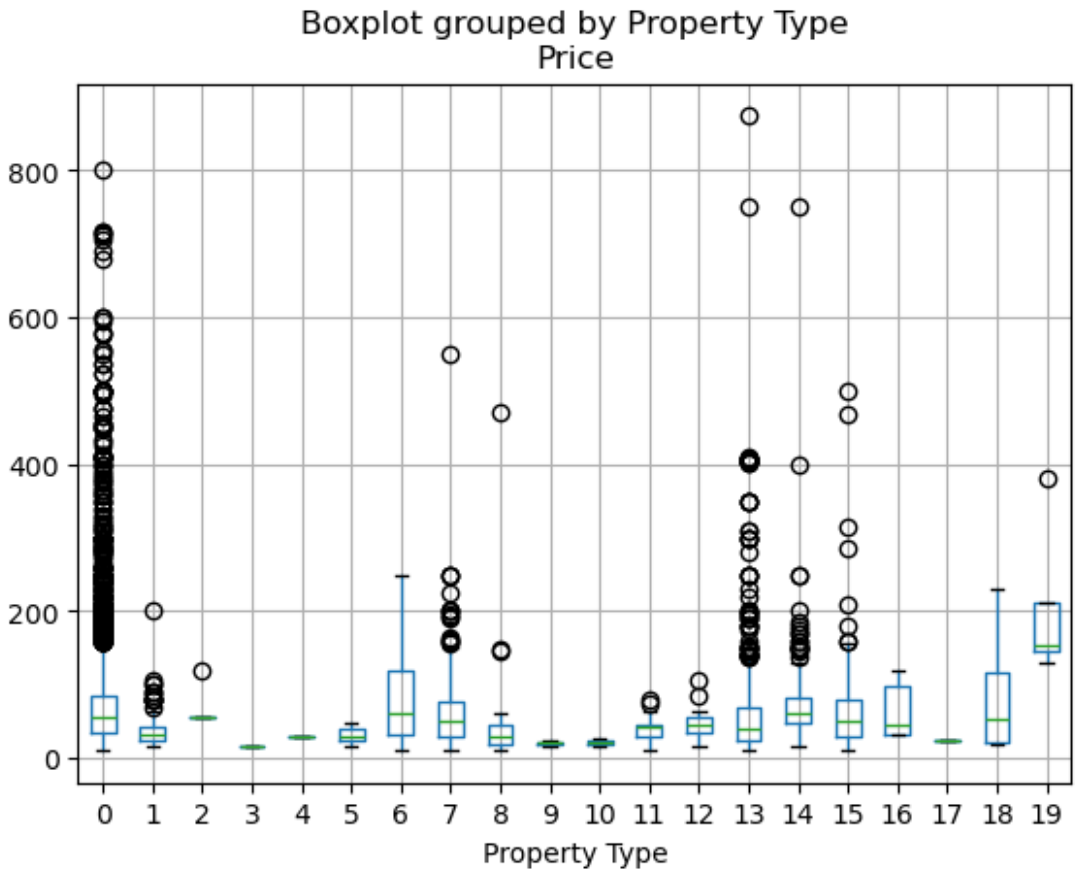

```
# Sólo representamos 3: bedrooms, sqm_living y waterfront
# el resto se puede repetir una a una

house_data.plot(kind = 'scatter',x='Bedrooms',y = 'Price')
plt.xlabel('Bedrooms')
plt.ylabel('Price (€)')
plt.show()

house_data.plot(kind = 'scatter',x='Accommodates',y = 'Price')
plt.xlabel('Accommodates')
plt.ylabel('Price (€)')
plt.show()
house_data.plot(kind = 'scatter',x='Number of Reviews',y = 'Price')
plt.xlabel('Number of Reviews')
plt.ylabel('Price (€)')
plt.show()

house_data.boxplot(by='Room Type',column = 'Price')
plt.show()

house_data.boxplot(by='Property Type',column = 'Price')
plt.show()
```


Price (€)
800
600
400
200
0
0
2
4
6
8
10
Bedrooms


Price (€)
800
600
400
200
0
2
4
6
8
10
12
14
16
Accommodates


800
600
400
200
0
Price (€)
0
50
100
150
200
250
300
350
Number of Reviews


Boxplot grouped by Room Type
Price
800
600
400
200
0
0
1
2
Room Type




```
# Categorías codificadas para 'Room Type':
#   Entire home/apt: 0
#   Private room: 1
#   Shared room: 2

# Categorías codificadas para 'Property Type':
#   Apartment: 0
#   Bed & Breakfast: 1
#   Boutique hotel: 2
#   Bungalow: 3
#   Camper/RV: 4
#   Casa particular: 5
#   Chalet: 6
#   Condominium: 7
#   Dorm: 8
#   Earth House: 9
#   Guest suite: 10
#   Guesthouse: 11
#   Hostel: 12
#   House: 13
#   Loft: 14
#   Other: 15
```

```
#   Serviced apartment: 16
#   Tent: 17
#   Townhouse: 18
#   Villa: 19

# Eliminación de outliers

property_categories = {
    0: "Apartment",
    1: "Bed & Breakfast",
    2: "Boutique Hotel",
    3: "Bungalow",
    4: "Camper/RV",
    5: "Casa Particular",
    6: "Chalet",
    7: "Condominium",
    8: "Dorm",
    9: "Earth House",
    10: "Guest Suite",
    11: "Guesthouse",
    12: "Hostel",
    13: "House",
    14: "Loft",
    15: "Other",
    16: "Serviced Apartment",
    17: "Tent",
    18: "Townhouse",
    19: "Villa"
}

# Crear un diccionario para almacenar los resultados
property_counts = {}

# Iterar por cada categoría y contar las ocurrencias en 'Property
Type'
for category_id, category_name in property_categories.items():
    count = house_data[house_data['Property Type'] ==
category_id].shape[0]
    property_counts[category_name] = count
    print(f"{category_name}: {count}")

# Opcional: Mostrar resultados ordenados
sorted_property_counts = sorted(property_counts.items(), key=lambda x:
x[1], reverse=True)
for category_name, count in sorted_property_counts:
    print(f"{category_name}: {count}")


Apartment: 8726
Bed & Breakfast: 256
```

```
Boutique Hotel: 5
Bungalow: 1
Camper/RV: 1
Casa Particular: 3
Chalet: 18
Condominium: 258
Dorm: 37
Earth House: 2
Guest Suite: 2
Guesthouse: 29
Hostel: 13
House: 774
Loft: 227
Other: 161
Serviced Apartment: 9
Tent: 1
Townhouse: 8
Villa: 4
Apartment: 8726
House: 774
Condominium: 258
Bed & Breakfast: 256
Loft: 227
Other: 161
Dorm: 37
Guesthouse: 29
Chalet: 18
Hostel: 13
Serviced Apartment: 9
Townhouse: 8
Boutique Hotel: 5
Villa: 4
Casa Particular: 3
Earth House: 2
Guest Suite: 2
Bungalow: 1
Camper/RV: 1
Tent: 1
```

```python
# Categorías a eliminar
categories_to_remove = [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18,
19]

# Filtrar el dataset para mantener solo las categorías relevantes
house_data_filtered = house_data[~house_data['Property
Type'].isin(categories_to_remove)]

# Verificar el resultado
print(house_data_filtered['Property Type'].value_counts())
```

```
Property Type
0     8726
13     774
1      256
14     227
11      29
12      13
Name: count, dtype: int64
```

```
outliers_extremos = house_data[
    (house_data['Bedrooms'] > 6) |
    (house_data['Bathrooms'] > 5) |
    (house_data['Beds'] > 10) |
    (house_data['Accommodates'] > 12)
]

print(outliers_extremos[['Property Type', 'Bedrooms', 'Bathrooms',
'Beds', 'Accommodates']])
```

	Property Type	Bedrooms	Bathrooms	Beds	Accommodates
181	12	1.000000	3.0	10.000000	16
295	19	8.000000	5.5	14.000000	16
500	0	4.000000	2.0	11.000000	14
855	13	1.000000	5.5	1.000000	3
1075	6	4.000000	3.0	12.000000	11
1240	19	4.000000	3.0	12.000000	12
1327	0	6.000000	2.0	16.000000	16
1460	0	4.000000	4.0	11.000000	16
1540	1	10.000000	6.0	16.000000	1
1663	0	7.000000	4.5	15.000000	16
1680	1	1.000000	6.0	1.000000	2
1882	0	1.000000	4.0	14.000000	16
1972	0	6.000000	6.0	2.000000	12
2398	0	5.000000	3.0	14.000000	16
2474	15	5.000000	5.0	8.000000	16
2553	0	5.000000	2.0	16.000000	16
2611	0	8.000000	3.0	13.000000	16
2699	1	1.000000	6.0	1.000000	1
2820	0	3.000000	2.0	12.000000	12
3097	0	3.000000	2.0	14.000000	14
3139	0	4.000000	4.0	8.000000	16
3812	0	1.000000	8.0	16.000000	16
3828	14	5.000000	4.0	9.000000	13
4142	12	1.000000	8.0	1.988951	10
4242	1	1.000000	6.0	1.000000	1
4350	13	10.000000	5.5	1.000000	1
4610	0	5.000000	5.0	10.000000	14
4682	0	5.000000	6.0	12.000000	12
4793	7	1.000000	2.0	13.000000	13
5306	1	1.000000	6.0	1.000000	2

```
5403                0   7.000000         3.0  16.000000            16
5774                0   8.000000         3.0  12.000000            16
5845                0   5.000000         5.5  10.000000            14
6349                0   5.000000         2.0  11.000000            16
6624                0   3.000000         3.5  10.000000            15
6818                0  10.000000         6.0  16.000000            16
6953               13   4.000000         6.0   9.000000            10
7072                2   1.293715         6.0   1.988951             2
7169               13   7.000000         6.0   9.000000            14
7329                1   1.000000         6.0   1.000000             1
7528               13  10.000000         0.0   2.000000             2
7725               14  10.000000         3.0  16.000000            16
8092                0   1.000000         1.0  12.000000            14
8209                0   4.000000         2.0  11.000000            12
8520                0   4.000000         2.0   5.000000            14
8660                0   6.000000         2.0   7.000000            16
8958                0   3.000000         2.0  10.000000            14
9018               13   1.000000         6.0   2.000000             2
9255                0   4.000000         2.0   8.000000            13
9422               12   1.000000         8.0  16.000000             2
9647                1   1.000000         6.0   1.000000             1
9715                0   5.000000         2.0  16.000000            16
9791               12   7.000000         7.0   7.000000             1
10286               0   8.000000         4.5  16.000000            16
10709               0   5.000000         2.0  10.000000            14
11030               6   6.000000         6.0  12.000000            12
11255               0   5.000000         5.0  13.000000            16
11352               0   6.000000         6.0  16.000000            16
11595               1   1.000000         6.0   1.000000             3
11703               0   1.000000         6.0  10.000000            10
```

```
# Filtrar propiedades con Availability en 0 y Número de Reviews
cercano a 0
availability_zero_reviews_low = house_data[
    ((house_data['Availability 30'] == 0) |
     (house_data['Availability 60'] == 0) |
     (house_data['Availability 90'] == 0) |
     (house_data['Availability 365'] == 0)) &
    (house_data['Number of Reviews'] <= 1)  # Ajusta el umbral según
sea necesario
]

# Ver el resultado
print(availability_zero_reviews_low.shape)
print(availability_zero_reviews_low[['Availability 30', 'Availability
60', 'Availability 90', 'Availability 365', 'Number of
Reviews']].head())
```

```
(1209, 22)
    Availability 30  Availability 60  Availability 90  Availability
```

```
365  \
3                   0                   6                  36
311
7                   0                   0                  18
74
11                  0                   0                   6
281
16                  0                   0                   0
0
33                  0                   6                  12
92

    Number of Reviews
3                   0
7                   0
11                  0
16                  0
33                  1
```

```python
# Filtrar propiedades con Availability en 0 en todas las columnas
fully_inactive_properties = house_data[
    (house_data['Availability 30'] == 0) &
    (house_data['Availability 60'] == 0) &
    (house_data['Availability 90'] == 0) &
    (house_data['Availability 365'] == 0)
]

# Eliminar estas propiedades del dataset original
house_data_filtered = house_data.drop(fully_inactive_properties.index)

# Verificar
print(f"Propiedades eliminadas: {fully_inactive_properties.shape[0]}")
print(f"Dataset resultante: {house_data_filtered.shape}")
```

```
Propiedades eliminadas: 893
Dataset resultante: (9642, 22)
```

```python
# Comparar el tamaño del dataset original con el modificado después de eliminar outliers e inactivos
print(
    f'Original: {house_data.shape[0]} // '
    f'Modificado: {house_data_filtered.shape[0]}\n'
    f'Diferencia: {house_data.shape[0] - house_data_filtered.shape[0]}'
)
print(
    f'Variación: {((house_data.shape[0] - house_data_filtered.shape[0]) / house_data.shape[0]) * 100:.2f}%'
)
```

```
Original: 10535 // Modificado: 9642
Diferencia: 893
Variación: 8.48%

house_data.corr() # matriz de correlación

                                Host Listings Count  Latitude
Longitude  \
Host Listings Count                        1.000000  0.014304  -
0.038058
Latitude                                   0.014304  1.000000
0.213476
Longitude                                 -0.038058  0.213476
1.000000
Accommodates                               0.126547  0.002178  -
0.044600
Guests Included                            0.054478 -0.013013  -
0.028390
Extra People                               0.020036 -0.002512  -
0.015830
Minimum Nights                             0.025171 -0.005330
0.000258
Maximum Nights                             0.004252  0.005138  -
0.002936
Availability 30                           -0.051244  0.069213
0.036139
Availability 60                           -0.059563  0.066229
0.035391
Availability 90                           -0.051332  0.058879
0.036191
Availability 365                           0.050653  0.037229  -
0.001553
Number of Reviews                         -0.094995 -0.070268  -
0.044190
Calculated host listings count             0.905276  0.014931  -
0.039239
Bathrooms                                  0.154126  0.038221
0.011891
Bedrooms                                   0.070887  0.025796
0.010724
Beds                                       0.082156  0.018611  -
0.018282
Price                                      0.164997  0.120396  -
0.041849
Room Type                                 -0.109603 -0.018962
0.045678
Property Type                             -0.085788  0.045008
0.036785
Cancellation Policy                        0.251032 -0.039154  -
0.039147
```

```
Neighbourhood Group Cleansed              -0.028009  0.183631
0.371983

                                Accommodates  Guests Included  Extra
People  \
Host Listings Count                 0.126547         0.054478
0.020036
Latitude                            0.002178        -0.013013     -
0.002512
Longitude                          -0.044600        -0.028390     -
0.015830
Accommodates                        1.000000         0.579907
0.279163
Guests Included                     0.579907         1.000000
0.357015
Extra People                        0.279163         0.357015
1.000000
Minimum Nights                      0.000847         0.001191     -
0.018471
Maximum Nights                      0.000799        -0.005224     -
0.002424
Availability 30                    -0.069867        -0.089394
0.038871
Availability 60                    -0.062581        -0.074709
0.051921
Availability 90                    -0.068686        -0.073119
0.052441
Availability 365                    0.079242         0.058600
0.102075
Number of Reviews                   0.086104         0.122976
0.073082
Calculated host listings count      0.129265         0.056002
0.018571
Bathrooms                           0.331659         0.194126
0.097281
Bedrooms                            0.673633         0.429732
0.149794
Beds                                0.821571         0.456095
0.217584
Price                               0.512650         0.320772
0.085962
Room Type                          -0.491595        -0.360986     -
0.106533
Property Type                      -0.052347        -0.029449     -
0.009350
Cancellation Policy                 0.220893         0.198636
0.156583
Neighbourhood Group Cleansed       -0.062832        -0.064583     -
0.052061
```

	Minimum Nights	Maximum Nights	\
Host Listings Count	0.025171	0.004252	
Latitude	-0.005330	0.005138	
Longitude	0.000258	-0.002936	
Accommodates	0.000847	0.000799	
Guests Included	0.001191	-0.005224	
Extra People	-0.018471	-0.002424	
Minimum Nights	1.000000	-0.001077	
Maximum Nights	-0.001077	1.000000	
Availability 30	-0.017176	0.002051	
Availability 60	-0.024450	0.000601	
Availability 90	-0.024035	-0.000700	
Availability 365	0.002812	0.010599	
Number of Reviews	-0.035536	-0.005655	
Calculated host listings count	0.027103	0.005305	
Bathrooms	0.025066	-0.003841	
Bedrooms	0.013644	-0.000012	
Beds	0.000246	0.000801	
Price	0.027671	0.002471	
Room Type	-0.027512	-0.007514	
Property Type	-0.003345	0.002314	
Cancellation Policy	0.033324	0.004191	
Neighbourhood Group Cleansed	0.019026	-0.003621	

	Availability 30	Availability 60	\
Host Listings Count	-0.051244	-0.059563	
Latitude	0.069213	0.066229	
Longitude	0.036139	0.035391	
Accommodates	-0.069867	-0.062581	
Guests Included	-0.089394	-0.074709	
Extra People	0.038871	0.051921	
Minimum Nights	-0.017176	-0.024450	
Maximum Nights	0.002051	0.000601	
Availability 30	1.000000	0.922544	
Availability 60	0.922544	1.000000	
Availability 90	0.852845	0.964059	
Availability 365	0.421739	0.485225	
Number of Reviews	-0.132393	-0.109170	
Calculated host listings count	-0.059044	-0.061903	
Bathrooms	-0.026658	-0.039622	
Bedrooms	-0.030745	-0.039708	
Beds	-0.005566	-0.007816	
Price	0.072327	0.049246	
Room Type	0.253943	0.233849	
Property Type	0.092665	0.079754	
Cancellation Policy	-0.099542	-0.062916	
Neighbourhood Group Cleansed	0.104606	0.087508	

	Availability 90	Availability 365	\

```
Host Listings Count                -0.051332          0.050653
Latitude                            0.058879          0.037229
Longitude                           0.036191         -0.001553
Accommodates                       -0.068686          0.079242
Guests Included                    -0.073119          0.058600
Extra People                        0.052441          0.102075
Minimum Nights                     -0.024035          0.002812
Maximum Nights                     -0.000700          0.010599
Availability 30                     0.852845          0.421739
Availability 60                     0.964059          0.485225
Availability 90                     1.000000          0.529154
Availability 365                    0.529154          1.000000
Number of Reviews                  -0.095881          0.082409
Calculated host listings count     -0.044463          0.085668
Bathrooms                          -0.039385          0.000122
Bedrooms                           -0.048823          0.017589
Beds                               -0.017420          0.086464
Price                               0.029831          0.082557
Room Type                           0.226258          0.018529
Property Type                       0.073702          0.027025
Cancellation Policy                -0.039047          0.102644
Neighbourhood Group Cleansed        0.077489          0.019604

                                Number of Reviews  \
Host Listings Count                     -0.094995
Latitude                                -0.070268
Longitude                               -0.044190
Accommodates                             0.086104
Guests Included                          0.122976
Extra People                             0.073082
Minimum Nights                          -0.035536
Maximum Nights                          -0.005655
Availability 30                         -0.132393
Availability 60                         -0.109170
Availability 90                         -0.095881
Availability 365                         0.082409
Number of Reviews                        1.000000
Calculated host listings count          -0.087071
Bathrooms                               -0.059843
Bedrooms                                -0.028202
Beds                                     0.037102
Price                                   -0.044685
Room Type                               -0.144229
Property Type                           -0.054610
Cancellation Policy                      0.174134
Neighbourhood Group Cleansed            -0.148705

                                Calculated host listings count
Bathrooms  \
```

```
Host Listings Count                                0.905276
0.154126
Latitude                                           0.014931
0.038221
Longitude                                         -0.039239
0.011891
Accommodates                                       0.129265
0.331659
Guests Included                                    0.056002
0.194126
Extra People                                       0.018571
0.097281
Minimum Nights                                     0.027103
0.025066
Maximum Nights                                     0.005305  -
0.003841
Availability 30                                   -0.059044  -
0.026658
Availability 60                                   -0.061903  -
0.039622
Availability 90                                   -0.044463  -
0.039385
Availability 365                                   0.085668
0.000122
Number of Reviews                                 -0.087071  -
0.059843
Calculated host listings count                     1.000000
0.175006
Bathrooms                                          0.175006
1.000000
Bedrooms                                           0.066327
0.421046
Beds                                               0.091991
0.386494
Price                                              0.136267
0.303140
Room Type                                         -0.089579
0.026760
Property Type                                     -0.088786
0.027827
Cancellation Policy                                0.256623
0.124406
Neighbourhood Group Cleansed                      -0.033380
0.006015

                                  Bedrooms      Beds     Price  Room
Type  \
Host Listings Count               0.070887  0.082156  0.164997  -
0.109603
```

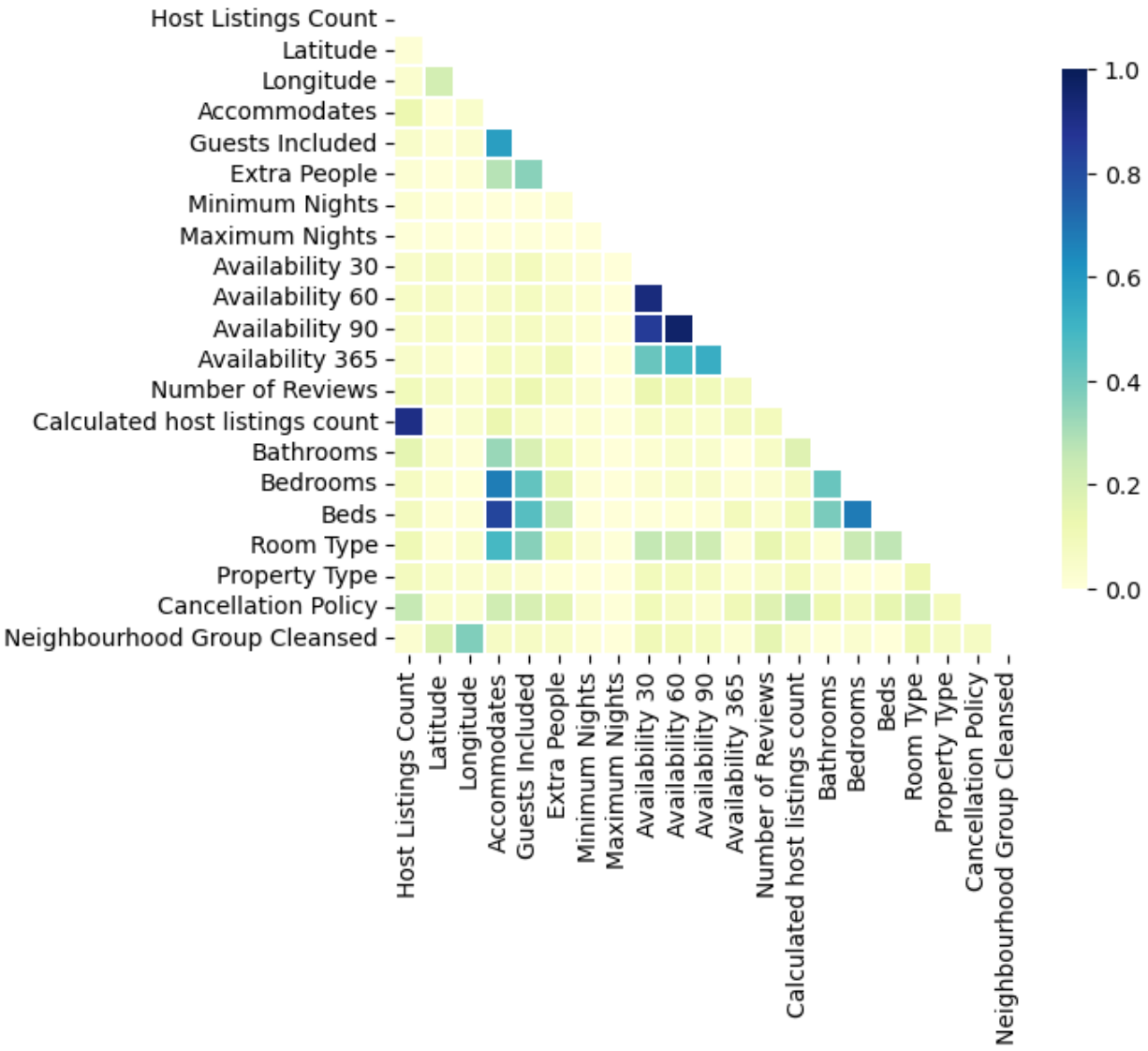
```
Latitude                          0.025796  0.018611  0.120396  -
0.018962
Longitude                         0.010724 -0.018282 -0.041849
0.045678
Accommodates                      0.673633  0.821571  0.512650  -
0.491595
Guests Included                   0.429732  0.456095  0.320772  -
0.360986
Extra People                      0.149794  0.217584  0.085962  -
0.106533
Minimum Nights                    0.013644  0.000246  0.027671  -
0.027512
Maximum Nights                   -0.000012  0.000801  0.002471  -
0.007514
Availability 30                  -0.030745 -0.005566  0.072327
0.253943
Availability 60                  -0.039708 -0.007816  0.049246
0.233849
Availability 90                  -0.048823 -0.017420  0.029831
0.226258
Availability 365                  0.017589  0.086464  0.082557
0.018529
Number of Reviews                -0.028202  0.037102 -0.044685  -
0.144229
Calculated host listings count    0.066327  0.091991  0.136267  -
0.089579
Bathrooms                         0.421046  0.386494  0.303140
0.026760
Bedrooms                          1.000000  0.679295  0.462199  -
0.240459
Beds                              0.679295  1.000000  0.420020  -
0.262747
Price                             0.462199  0.420020  1.000000  -
0.388079
Room Type                        -0.240459 -0.262747 -0.388079
1.000000
Property Type                     0.009896 -0.002595 -0.001406
0.120949
Cancellation Policy               0.085865  0.146140  0.099008  -
0.203567
Neighbourhood Group Cleansed      0.033887 -0.001879 -0.011183
0.112046

                                  Property Type  Cancellation Policy  \
Host Listings Count                   -0.085788             0.251032
Latitude                               0.045008            -0.039154
Longitude                              0.036785            -0.039147
Accommodates                          -0.052347             0.220893
Guests Included                       -0.029449             0.198636
```

Extra People	-0.009350	0.156583
Minimum Nights	-0.003345	0.033324
Maximum Nights	0.002314	0.004191
Availability 30	0.092665	-0.099542
Availability 60	0.079754	-0.062916
Availability 90	0.073702	-0.039047
Availability 365	0.027025	0.102644
Number of Reviews	-0.054610	0.174134
Calculated host listings count	-0.088786	0.256623
Bathrooms	0.027827	0.124406
Bedrooms	0.009896	0.085865
Beds	-0.002595	0.146140
Price	-0.001406	0.099008
Room Type	0.120949	-0.203567
Property Type	1.000000	-0.088810
Cancellation Policy	-0.088810	1.000000
Neighbourhood Group Cleansed	0.069363	-0.068839

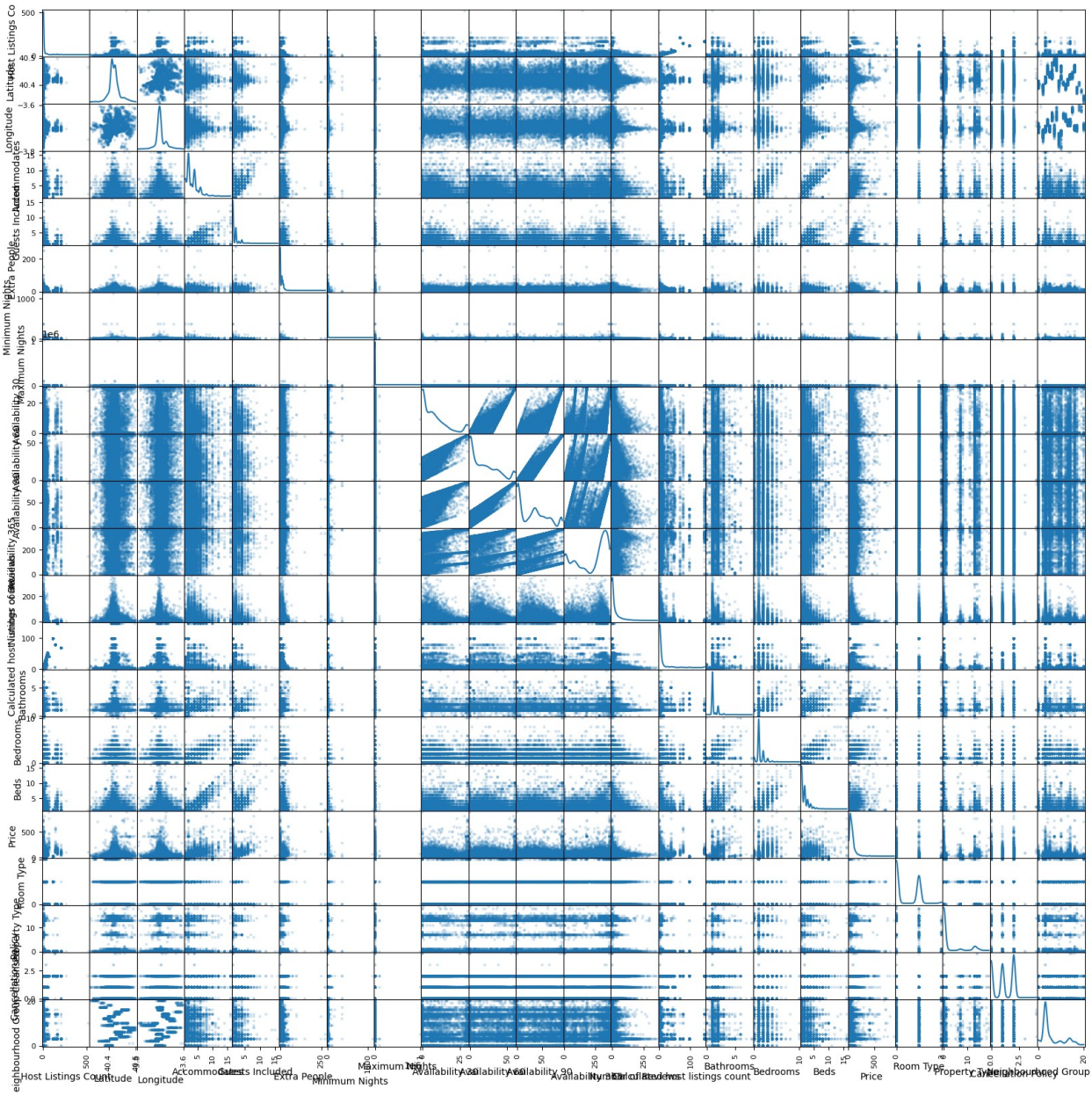
	Neighbourhood Group Cleansed
Host Listings Count	-0.028009
Latitude	0.183631
Longitude	0.371983
Accommodates	-0.062832
Guests Included	-0.064583
Extra People	-0.052061
Minimum Nights	0.019026
Maximum Nights	-0.003621
Availability 30	0.104606
Availability 60	0.087508
Availability 90	0.077489
Availability 365	0.019604
Number of Reviews	-0.148705
Calculated host listings count	-0.033380
Bathrooms	0.006015
Bedrooms	0.033887
Beds	-0.001879
Price	-0.011183
Room Type	0.112046
Property Type	0.069363
Cancellation Policy	-0.068839
Neighbourhood Group Cleansed	1.000000

```
import seaborn as sns

# Compute the correlation matrix
corr = np.abs(house_data.drop(['Price'], axis=1).corr())

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=bool)
mask[np.triu_indices_from(mask)] = True
```

```
# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(6, 5))

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,vmin = 0.0, vmax=1.0, center=0.5,
            linewidths=.1, cmap="YlGnBu", cbar_kws={"shrink": .8})

plt.show()
```




```
pd.plotting.scatter_matrix(house_data, alpha=0.2, figsize=(20, 20),
diagonal = 'kde')
plt.show()
```




```
# Combina Bedrooms, Beds y Accommodates en una métrica única para representar la capacidad total ajustada por características de la propiedad:
house_data['Adjusted Capacity'] = house_data['Accommodates'] + 0.5 * house_data['Beds'] + 0.25 * house_data['Bedrooms']

# Relaciona la actividad de la propiedad (Number of Reviews) con su disponibilidad total. Esto podría ser útil para identificar propiedades más populares o mal registradas:
house_data['Review Ratio'] = house_data['Number of Reviews'] / (house_data['Availability 365'] + 1)

# Crea una métrica que capture el precio relativo de la propiedad por
```

```
persona que puede alojar:
house_data['Price per Person'] = house_data['Price'] /
house_data['Accommodates']

# Para variables categóricas como Property Type y Room Type, podrías
generar promedios de precio por categoría y usarlos como nuevas
características:
house_data['Avg Price by Room Type'] = house_data.groupby('Room Type')
['Price'].transform('mean')
house_data['Avg Price by Property Type'] =
house_data.groupby('Property Type')['Price'].transform('mean')


# Eliminar columnas de disponibilidad innecesarias
house_data = house_data.drop(columns=['Availability 30', 'Availability
60', 'Availability 90'])

# Renombrar Availability 365 (opcional)
house_data = house_data.rename(columns={'Availability 365':
'Availability'})

# Verificar los cambios
print(house_data.columns)
```

```
Index(['Host Listings Count', 'Latitude', 'Longitude', 'Accommodates',
       'Guests Included', 'Extra People', 'Minimum Nights', 'Maximum
Nights',
       'Availability', 'Number of Reviews', 'Calculated host listings
count',
       'Bathrooms', 'Bedrooms', 'Beds', 'Price', 'Room Type',
'Property Type',
       'Cancellation Policy', 'Neighbourhood Group Cleansed',
       'Adjusted Capacity', 'Review Ratio', 'Price per Person',
       'Avg Price by Room Type', 'Avg Price by Property Type'],
      dtype='object')
```

```
Modelado, cross-validation y estudio de resultados en train y test
```

```
  Cell In[32], line 1
    Modelado, cross-validation y estudio de resultados en train y test
                                ^
SyntaxError: invalid syntax
```

```
# Carga de datos
house_data = pd.read_csv('./Practica/airbnb_train.csv', sep=';',
decimal='.')

# Filtrar solo propiedades en Madrid
house_data = house_data[
    (house_data['Latitude'] >= 40.3) & (house_data['Latitude'] <=
```

```
40.5) &
    (house_data['Longitude'] >= -3.8) & (house_data['Longitude'] <= -
3.6)
]

# Imputación
columns_with_mode = ['Room Type', 'Property Type', 'Cancellation
Policy']
for col in columns_with_mode:
    house_data[col] = house_data[col].fillna(house_data[col].mode()
[0])

# Imputar con la media para columnas numéricas
columns_with_mean = ['Price', 'Bathrooms', 'Bedrooms', 'Beds']
for col in columns_with_mean:
    house_data[col] = house_data[col].fillna(house_data[col].mean())

# Imputar con la mediana para otras columnas numéricas
columns_with_median = ['Minimum Nights', 'Maximum Nights',
'Availability 30']
for col in columns_with_median:
    house_data[col] = house_data[col].fillna(house_data[col].median())

from sklearn.preprocessing import LabelEncoder

# Definir las columnas categóricas a codificar
columns_label_encode = ['Room Type', 'Property Type', 'Cancellation
Policy', 'Neighbourhood Group Cleansed']

# Inicializar el LabelEncoder
label_encoders = {}  # Guardar los encoders para cada columna

# Aplicar LabelEncoder a cada columna
for column in columns_label_encode:
    le = LabelEncoder()
    house_data[column] = le.fit_transform(house_data[column])
    label_encoders[column] = le  # Guardar el encoder para referencia
futura

# Ver los valores codificados y sus significados
for column, le in label_encoders.items():
    print(f"Categorías codificadas para '{column}':")
    for class_, encoded_value in zip(le.classes_,
le.transform(le.classes_)):
        print(f"  {class_}: {encoded_value}")
    print("\n")


house_data['Host Listings Count'].fillna(house_data['Host Listings
Count'].mode()[0], inplace=True)
```

```python
# Eliminamos las columnas
columns_to_keep = [
    'Host Listings Count', 'Latitude', 'Longitude', 'Accommodates',
'Guests Included',
    'Extra People', 'Minimum Nights', 'Maximum Nights', 'Availability
30',
    'Availability 60', 'Availability 90', 'Availability 365', 'Number
of Reviews',
    'Calculated host listings count', 'Bathrooms', 'Bedrooms', 'Beds',
'Price',  # Columna objetivo
    'Room Type', 'Property Type', 'Cancellation Policy',
'Neighbourhood Group Cleansed'
]

# Eliminar todas las columnas que no estén en la lista
house_data = house_data[columns_to_keep]

# Eliminamos outliers en bedrooms
property_categories = {
    0: "Apartment",
    1: "Bed & Breakfast",
    2: "Boutique Hotel",
    3: "Bungalow",
    4: "Camper/RV",
    5: "Casa Particular",
    6: "Chalet",
    7: "Condominium",
    8: "Dorm",
    9: "Earth House",
    10: "Guest Suite",
    11: "Guesthouse",
    12: "Hostel",
    13: "House",
    14: "Loft",
    15: "Other",
    16: "Serviced Apartment",
    17: "Tent",
    18: "Townhouse",
    19: "Villa"
}

# Crear un diccionario para almacenar los resultados
property_counts = {}

# Iterar por cada categoría y contar las ocurrencias en 'Property
Type'
for category_id, category_name in property_categories.items():
    count = house_data[house_data['Property Type'] ==
category_id].shape[0]
```

```
    property_counts[category_name] = count
    print(f"{category_name}: {count}")

# Categorías a eliminar
categories_to_remove = [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18,
19]

# Filtrar el dataset para mantener solo las categorías relevantes
house_data_filtered = house_data[~house_data['Property
Type'].isin(categories_to_remove)]

outliers_extremos = house_data[
    (house_data['Bedrooms'] > 6) |
    (house_data['Bathrooms'] > 5) |
    (house_data['Beds'] > 10) |
    (house_data['Accommodates'] > 12)
]

# Filtrar propiedades con Availability en 0 en todas las columnas
fully_inactive_properties = house_data[
    (house_data['Availability 30'] == 0) &
    (house_data['Availability 60'] == 0) &
    (house_data['Availability 90'] == 0) &
    (house_data['Availability 365'] == 0)
]

# Eliminar estas propiedades del dataset original
house_data_filtered = house_data.drop(fully_inactive_properties.index)

# Generamos características
# Combina Bedrooms, Beds y Accommodates en una métrica única para
representar la capacidad total ajustada por características de la
propiedad:
house_data['Adjusted Capacity'] = house_data['Accommodates'] + 0.5 *
house_data['Beds'] + 0.25 * house_data['Bedrooms']

# Relaciona la actividad de la propiedad (Number of Reviews) con su
disponibilidad total. Esto podría ser útil para identificar
propiedades más populares o mal registradas:
house_data['Review Ratio'] = house_data['Number of Reviews'] /
(house_data['Availability 365'] + 1)

# Crea una métrica que capture el precio relativo de la propiedad por
persona que puede alojar:
house_data['Price per Person'] = house_data['Price'] /
house_data['Accommodates']

# Para variables categóricas como Property Type y Room Type, podrías
generar promedios de precio por categoría y usarlos como nuevas
características:
```

```
house_data['Avg Price by Room Type'] = house_data.groupby('Room Type')
['Price'].transform('mean')
house_data['Avg Price by Property Type'] =
house_data.groupby('Property Type')['Price'].transform('mean')

# Eliminar columnas de disponibilidad innecesarias
house_data = house_data.drop(columns=['Availability 30', 'Availability
60', 'Availability 90'])

# Renombrar Availability 365 (opcional)
house_data = house_data.rename(columns={'Availability 365':
'Availability'})
```

```
Categorías codificadas para 'Room Type':
  Entire home/apt: 0
  Private room: 1
  Shared room: 2


Categorías codificadas para 'Property Type':
  Apartment: 0
  Bed & Breakfast: 1
  Boutique hotel: 2
  Bungalow: 3
  Camper/RV: 4
  Casa particular: 5
  Chalet: 6
  Condominium: 7
  Dorm: 8
  Earth House: 9
  Guest suite: 10
  Guesthouse: 11
  Hostel: 12
  House: 13
  Loft: 14
  Other: 15
  Serviced apartment: 16
  Tent: 17
  Townhouse: 18
  Villa: 19


Categorías codificadas para 'Cancellation Policy':
  flexible: 0
  moderate: 1
  strict: 2
  super_strict_30: 3
  super_strict_60: 4
```

```
Categorías codificadas para 'Neighbourhood Group Cleansed':
  Arganzuela: 0
  Barajas: 1
  Carabanchel: 2
  Centro: 3
  Chamartín: 4
  Chamberí: 5
  Ciudad Lineal: 6
  Fuencarral - El Pardo: 7
  Hortaleza: 8
  Latina: 9
  Moncloa - Aravaca: 10
  Moratalaz: 11
  Puente de Vallecas: 12
  Retiro: 13
  Salamanca: 14
  San Blas - Canillejas: 15
  Tetuán: 16
  Usera: 17
  Vicálvaro: 18
  Villa de Vallecas: 19
  Villaverde: 20


Apartment: 8726
Bed & Breakfast: 256
Boutique Hotel: 5
Bungalow: 1
Camper/RV: 1
Casa Particular: 3
Chalet: 18
Condominium: 258
Dorm: 37
Earth House: 2
Guest Suite: 2
Guesthouse: 29
Hostel: 13
House: 774
Loft: 227
Other: 161
Serviced Apartment: 9
Tent: 1
Townhouse: 8
Villa: 4

# Carga de datos
house_data_test = pd.read_csv('./Practica/airbnb_test.csv', sep=';',
decimal='.')

# Filtrar solo propiedades en Madrid
```

```
house_data_test = house_data_test[
    (house_data_test['Latitude'] >= 40.3) &
(house_data_test['Latitude'] <= 40.5) &
    (house_data_test['Longitude'] >= -3.8) &
(house_data_test['Longitude'] <= -3.6)
]

# Imputación
columns_with_mode = ['Room Type', 'Property Type', 'Cancellation
Policy']
for col in columns_with_mode:
    house_data_test[col] =
house_data_test[col].fillna(house_data_test[col].mode()[0])

# Imputar con la media para columnas numéricas
columns_with_mean = ['Price', 'Bathrooms', 'Bedrooms', 'Beds']
for col in columns_with_mean:
    house_data_test[col] =
house_data_test[col].fillna(house_data_test[col].mean())

# Imputar con la mediana para otras columnas numéricas
columns_with_median = ['Minimum Nights', 'Maximum Nights',
'Availability 30']
for col in columns_with_median:
    house_data_test[col] =
house_data_test[col].fillna(house_data_test[col].median())

from sklearn.preprocessing import LabelEncoder

# Definir las columnas categóricas a codificar
columns_label_encode = ['Room Type', 'Property Type', 'Cancellation
Policy', 'Neighbourhood Group Cleansed']

# Inicializar el LabelEncoder
label_encoders = {}  # Guardar los encoders para cada columna

# Aplicar LabelEncoder a cada columna
for column in columns_label_encode:
    le = LabelEncoder()
    house_data_test[column] =
le.fit_transform(house_data_test[column])
    label_encoders[column] = le  # Guardar el encoder para referencia
futura

# Ver los valores codificados y sus significados
for column, le in label_encoders.items():
    print(f"Categorías codificadas para '{column}':")
    for class_, encoded_value in zip(le.classes_,
le.transform(le.classes_)):
        print(f"  {class_}: {encoded_value}")
```

```
    print("\n")


house_data_test['Host Listings Count'].fillna(house_data_test['Host
Listings Count'].mode()[0], inplace=True)

# Eliminamos las columnas
columns_to_keep = [
    'Host Listings Count', 'Latitude', 'Longitude', 'Accommodates',
'Guests Included',
    'Extra People', 'Minimum Nights', 'Maximum Nights', 'Availability
30',
    'Availability 60', 'Availability 90', 'Availability 365', 'Number
of Reviews',
    'Calculated host listings count', 'Bathrooms', 'Bedrooms', 'Beds',
'Price',  # Columna objetivo
    'Room Type', 'Property Type', 'Cancellation Policy',
'Neighbourhood Group Cleansed'
]

# Eliminar todas las columnas que no estén en la lista
house_data_test = house_data_test[columns_to_keep]

# Eliminamos outliers en bedrooms
property_categories = {
    0: "Apartment",
    1: "Bed & Breakfast",
    2: "Boutique Hotel",
    3: "Bungalow",
    4: "Camper/RV",
    5: "Casa Particular",
    6: "Chalet",
    7: "Condominium",
    8: "Dorm",
    9: "Earth House",
    10: "Guest Suite",
    11: "Guesthouse",
    12: "Hostel",
    13: "House",
    14: "Loft",
    15: "Other",
    16: "Serviced Apartment",
    17: "Tent",
    18: "Townhouse",
    19: "Villa"
}

# Crear un diccionario para almacenar los resultados
property_counts = {}
```

```
# Iterar por cada categoría y contar las ocurrencias en 'Property
Type'
for category_id, category_name in property_categories.items():
    count = house_data_test[house_data_test['Property Type'] ==
category_id].shape[0]
    property_counts[category_name] = count
    print(f"{category_name}: {count}")

# Categorías a eliminar
categories_to_remove = [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18,
19]

# Filtrar el dataset para mantener solo las categorías relevantes
house_data_test_filtered = house_data_test[~house_data_test['Property
Type'].isin(categories_to_remove)]

outliers_extremos = house_data_test[
    (house_data_test['Bedrooms'] > 6) |
    (house_data_test['Bathrooms'] > 5) |
    (house_data_test['Beds'] > 10) |
    (house_data_test['Accommodates'] > 12)
]

# Filtrar propiedades con Availability en 0 en todas las columnas
fully_inactive_properties = house_data_test[
    (house_data_test['Availability 30'] == 0) &
    (house_data_test['Availability 60'] == 0) &
    (house_data_test['Availability 90'] == 0) &
    (house_data_test['Availability 365'] == 0)
]

# Eliminar estas propiedades del dataset original
house_data_test_filtered =
house_data_test.drop(fully_inactive_properties.index)

# Generamos características
# Combina Bedrooms, Beds y Accommodates en una métrica única para
representar la capacidad total ajustada por características de la
propiedad:
house_data_test['Adjusted Capacity'] = house_data_test['Accommodates']
+ 0.5 * house_data_test['Beds'] + 0.25 * house_data_test['Bedrooms']

# Relaciona la actividad de la propiedad (Number of Reviews) con su
disponibilidad total. Esto podría ser útil para identificar
propiedades más populares o mal registradas:
house_data_test['Review Ratio'] = house_data_test['Number of Reviews']
/ (house_data_test['Availability 365'] + 1)

# Crea una métrica que capture el precio relativo de la propiedad por
persona que puede alojar:
```

```
house_data_test['Price per Person'] = house_data_test['Price'] /
house_data_test['Accommodates']

# Para variables categóricas como Property Type y Room Type, podrías
generar promedios de precio por categoría y usarlos como nuevas
características:
house_data_test['Avg Price by Room Type'] =
house_data_test.groupby('Room Type')['Price'].transform('mean')
house_data_test['Avg Price by Property Type'] =
house_data_test.groupby('Property Type')['Price'].transform('mean')

# Eliminar columnas de disponibilidad innecesarias
house_data_test = house_data_test.drop(columns=['Availability 30',
'Availability 60', 'Availability 90'])

# Renombrar Availability 365 (opcional)
house_data_test = house_data_test.rename(columns={'Availability 365':
'Availability'})
```

```
Categorías codificadas para 'Room Type':
  Entire home/apt: 0
  Private room: 1
  Shared room: 2


Categorías codificadas para 'Property Type':
  Apartment: 0
  Bed & Breakfast: 1
  Boutique hotel: 2
  Camper/RV: 3
  Chalet: 4
  Condominium: 5
  Dorm: 6
  Earth House: 7
  Guest suite: 8
  Guesthouse: 9
  Hostel: 10
  House: 11
  Loft: 12
  Other: 13
  Serviced apartment: 14
  Timeshare: 15
  Townhouse: 16


Categorías codificadas para 'Cancellation Policy':
  flexible: 0
  moderate: 1
  strict: 2
```

```
Categorías codificadas para 'Neighbourhood Group Cleansed':
  Arganzuela: 0
  Carabanchel: 1
  Centro: 2
  Chamartín: 3
  Chamberí: 4
  Ciudad Lineal: 5
  Fuencarral - El Pardo: 6
  Hortaleza: 7
  Latina: 8
  Moncloa - Aravaca: 9
  Moratalaz: 10
  Puente de Vallecas: 11
  Retiro: 12
  Salamanca: 13
  San Blas - Canillejas: 14
  Tetuán: 15
  Usera: 16
  Vicálvaro: 17
  Villa de Vallecas: 18
  Villaverde: 19


Apartment: 2151
Bed & Breakfast: 73
Boutique Hotel: 1
Bungalow: 1
Camper/RV: 3
Casa Particular: 75
Chalet: 5
Condominium: 1
Dorm: 1
Earth House: 10
Guest Suite: 3
Guesthouse: 206
Hostel: 52
House: 54
Loft: 4
Other: 1
Serviced Apartment: 1
Tent: 0
Townhouse: 0
Villa: 0
```

```python
from sklearn import preprocessing

# Dataset de train
data_train = house_data.values
y_train = data_train[:,0:1]     # nos quedamos con la 1ª columna,
```

```
price
X_train = data_train[:,1:]      # nos quedamos con el resto

# Dataset de test
data_test = house_data_test.values
y_test = data_test[:,0:1]     # nos quedamos con la 1ª columna, price
X_test = data_test[:,1:]      # nos quedamos con el resto

# Escalamos (con los datos de train)
scaler = preprocessing.StandardScaler().fit(X_train)
XtrainScaled = scaler.transform(X_train)

# recordad que esta normalización/escalado la realizo con el scaler
anterior, basado en los datos de training!
XtestScaled = scaler.transform(X_test)

print('Datos entrenamiento: ', XtrainScaled.shape)
print('Datos test: ', XtestScaled.shape)
```

```
Datos entrenamiento:  (10535, 23)
Datos test:  (2642, 23)
```

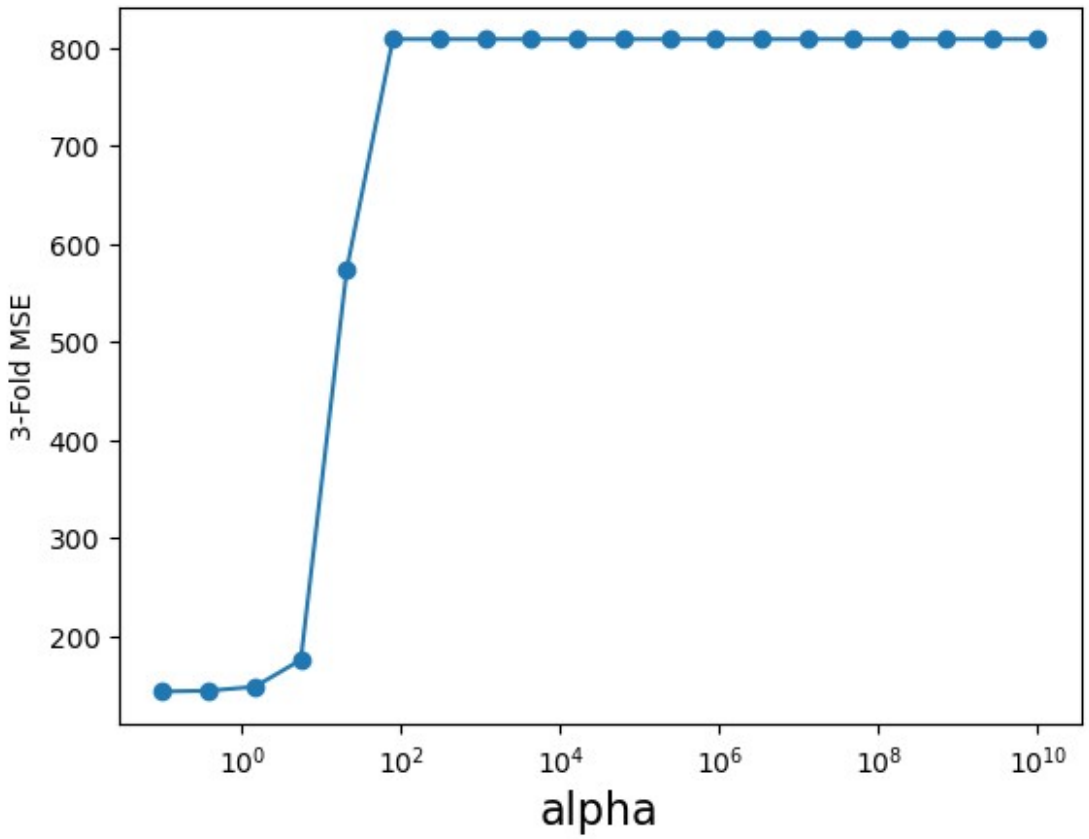
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso

alpha_vector = np.logspace(-1,10,20)
param_grid = {'alpha': alpha_vector }
grid = GridSearchCV(Lasso(), scoring= 'neg_mean_squared_error',
param_grid=param_grid, cv = 3, verbose=2)
grid.fit(XtrainScaled, y_train)
print("best mean cross-validation score:
{:.3f}".format(grid.best_score_))
print("best parameters: {}".format(grid.best_params_))

#-1 porque es negado
scores = -1*np.array(grid.cv_results_['mean_test_score'])
plt.semilogx(alpha_vector,scores,'-o')
plt.xlabel('alpha',fontsize=16)
plt.ylabel('3-Fold MSE')
plt.show()
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
[CV] END .....................................alpha=0.1; total
time=   3.6s
[CV] END .....................................alpha=0.1; total
time=   2.5s
[CV] END .....................................alpha=0.1; total
time=   1.1s
[CV] END ....................alpha=0.37926901907322497; total
time=   0.0s
[CV] END ....................alpha=0.37926901907322497; total
```

```
time=   0.9s
[CV] END .....................alpha=0.37926901907322497; total
time=   0.0s
[CV] END .....................alpha=1.438449888287663; total
time=   1.4s
[CV] END .....................alpha=1.438449888287663; total
time=   0.9s
[CV] END .....................alpha=1.438449888287663; total
time=   0.3s
[CV] END .....................alpha=5.45559478116852; total
time=   0.0s
[CV] END .....................alpha=5.45559478116852; total
time=   0.5s
[CV] END .....................alpha=5.45559478116852; total
time=   0.2s
[CV] END .....................alpha=20.6913808111479; total
time=   0.0s
[CV] END .....................alpha=20.6913808111479; total
time=   0.2s
[CV] END .....................alpha=20.6913808111479; total
time=   0.0s
[CV] END .....................alpha=78.47599703514615; total
time=   0.2s
[CV] END .....................alpha=78.47599703514615; total
time=   0.0s
[CV] END .....................alpha=78.47599703514615; total
time=   0.0s
[CV] END .....................alpha=297.63514416313194; total
time=   0.0s
[CV] END .....................alpha=297.63514416313194; total
time=   0.0s
[CV] END .....................alpha=297.63514416313194; total
time=   0.0s
[CV] END .....................alpha=1128.8378916846884; total
time=   0.0s
[CV] END .....................alpha=1128.8378916846884; total
time=   0.0s
[CV] END .....................alpha=1128.8378916846884; total
time=   0.0s
[CV] END .....................alpha=4281.332398719395; total
time=   0.0s
[CV] END .....................alpha=4281.332398719395; total
time=   0.0s
[CV] END .....................alpha=4281.332398719395; total
time=   0.1s
[CV] END .....................alpha=16237.767391887242; total
time=   0.0s
[CV] END .....................alpha=16237.767391887242; total
time=   0.0s
```

```
[CV] END .........................alpha=16237.767391887242; total
time=   0.0s
[CV] END .........................alpha=61584.82110660267; total
time=   0.0s
[CV] END .........................alpha=61584.82110660267; total
time=   0.1s
[CV] END .........................alpha=61584.82110660267; total
time=   0.0s
[CV] END .........................alpha=233572.14690901214; total
time=   0.0s
[CV] END .........................alpha=233572.14690901214; total
time=   0.0s
[CV] END .........................alpha=233572.14690901214; total
time=   0.0s
[CV] END .........................alpha=885866.7904100833; total
time=   0.0s
[CV] END .........................alpha=885866.7904100833; total
time=   0.0s
[CV] END .........................alpha=885866.7904100833; total
time=   0.0s
[CV] END .........................alpha=3359818.286283788; total
time=   0.0s
[CV] END .........................alpha=3359818.286283788; total
time=   0.0s
[CV] END .........................alpha=3359818.286283788; total
time=   0.1s
[CV] END .........................alpha=12742749.857031321; total
time=   0.0s
[CV] END .........................alpha=12742749.857031321; total
time=   0.0s
[CV] END .........................alpha=12742749.857031321; total
time=   0.0s
[CV] END .........................alpha=48329302.38571752; total
time=   0.1s
[CV] END .........................alpha=48329302.38571752; total
time=   0.0s
[CV] END .........................alpha=48329302.38571752; total
time=   0.0s
[CV] END .........................alpha=183298071.08324376; total
time=   0.0s
[CV] END .........................alpha=183298071.08324376; total
time=   0.0s
[CV] END .........................alpha=183298071.08324376; total
time=   0.2s
[CV] END .........................alpha=695192796.177562; total
time=   0.0s
[CV] END .........................alpha=695192796.177562; total
time=   0.0s
[CV] END .........................alpha=695192796.177562; total
```

```
time=   0.0s
[CV] END ..........................alpha=2636650898.730366; total
time=   0.0s
[CV] END ..........................alpha=2636650898.730366; total
time=   0.0s
[CV] END ..........................alpha=2636650898.730366; total
time=   0.0s
[CV] END ..............................alpha=10000000000.0; total
time=   0.0s
[CV] END ..............................alpha=10000000000.0; total
time=   0.0s
[CV] END ..............................alpha=10000000000.0; total
time=   0.0s
best mean cross-validation score: -143.921
best parameters: {'alpha': 0.1}
```




```python
from sklearn.metrics import mean_squared_error

alpha_optimo = grid.best_params_['alpha']
lasso = Lasso(alpha = alpha_optimo).fit(XtrainScaled,y_train)

ytrainLasso = lasso.predict(XtrainScaled)
ytestLasso  = lasso.predict(XtestScaled)
```

```
mseTrainModelLasso = mean_squared_error(y_train,ytrainLasso)
mseTestModelLasso = mean_squared_error(y_test,ytestLasso)

print('MSE Modelo Lasso (train): %0.3g' % mseTrainModelLasso)
print('MSE Modelo Lasso (test) : %0.3g' % mseTestModelLasso)

print('RMSE Modelo Lasso (train): %0.3g' %
np.sqrt(mseTrainModelLasso))
print('RMSE Modelo Lasso (test) : %0.3g' % np.sqrt(mseTestModelLasso))

feature_names = house_data.columns[1:] # es igual en train y en test

w = lasso.coef_
for f,wi in zip(feature_names,w):
    print(f,wi)
```

```
MSE Modelo Lasso (train): 143
MSE Modelo Lasso (test) : 82.4
RMSE Modelo Lasso (train): 11.9
RMSE Modelo Lasso (test) : 9.08
Latitude -0.0
Longitude -0.0
Accommodates -0.0
Guests Included -0.09104001985277002
Extra People 0.00988098094115094
Minimum Nights -0.0
Maximum Nights -0.0
Availability -0.7359492331270471
Number of Reviews -0.4011851322511225
Calculated host listings count 25.40926007248211
Bathrooms -0.2707627713834555
Bedrooms 0.0
Beds -0.38561817446235647
Price 1.2642990396959237
Room Type -0.08656212738170542
Property Type -0.0
Cancellation Policy 0.5812730982648898
Neighbourhood Group Cleansed 0.0
Adjusted Capacity -0.0
Review Ratio -0.05872423233537992
Price per Person -0.0
Avg Price by Room Type 0.2208732176427459
Avg Price by Property Type -0.0
```