Practica Final

June 2, 2022

#

PROYECTO FINAL MACHINE LEARNING AIRBNB

Podrá ver nuestro progreso en https://github.com/AlvaroML95/PRACTICA_FINAL.git

PARTICIPANTES:

@bchamba314:BRYAM DAVID CHAMBA JARAMILLO

@MariaGR4:MARÍA GARCÍA RODRIGUEZ

@nothng-k:MANUEL GRAU ROLDÁN

@AlvaroML95:ÁLVARO MARTÍN LÓPEZ

@lnatesGH:LUIS JAVIER NATES ARECHAVALETA

@alvblas:ALVAR YEBEL DE BLAS FERNÁNDEZ

[1]:
```
#import funciones_auxiliares
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import datetime
#Análisis Exploratorio
from scipy import stats
from scipy.stats import norm
import inspect
#Aprendizaje automático
from sklearn.impute import SimpleImputer
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
import category_encoders as ce
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from sklearn.metrics import \
    explained_variance_score, mean_squared_error,\
    r2_score,mean_absolute_error
```

```
from sklearn.linear_model import LassoCV
from sklearn.model_selection import RepeatedKFold
from sklearn.svm import SVR
```

0.1 Introducción y Motivación

Como práctica final para este cuso de Aprendizaje automático hemos decidido desarrollar un evaluador automático de precios a partir de las carácterísticas de los alojamientos de Airbnb. Esta compañía es una plataforma con gran interés en hacer que los anfitriones capten la atención de clientes potenciales, pues de ahí deriva el beneficio económico de la empresa. El precio es un factor principal que influye de manera sustancial en la decisión de reserva por parte de clientes potenciales, por eso es importante que los anfitriones sepan cuál es el mejor precio para sus ofertas en función de sus características y los intereses de los clientes.

Este evaluador va a centrarse en qué características potenciales se correlacionan mejor con el precio de un alojamiento de Airbnb para que los anfitriones puedan ser aconsejados correctamente y elijan sus precios con mayor precisión. Al igual que en la práctica anterior, en este proyecto vamos a usar un conjunto de datos reales y por tanto potencialmente útiles a la hora de prepararnos para aplicar en un futuro las habilidades adquiridas durante este curso en el entorno laboral.

Para resumir las diferentes fases del trabajo, en un primer bloque de preprocessing analizaremos la estructura de los datos, imputaremos y trataremos outliers junto a datos faltantes, estudiaremos la variable objetivo "Price" (Precio del alojamiento), realizaremos varios gráficos para comprender mejor el dataset, estudiaremos la correlación entre variables y transformaremos y/o crearemos variables para mejorar el dataset.

Posteriormente tras la fase de preproceso, procederemos a preparar los datos de entrenamiento y test para comenzar a utilizar algoritmos estudiados en esta asignatura a fin de producir varios modelos de regresión de aprendizaje automático y calcular sus métricas. Nuestra elección en este caso han sido KNN, Regresión Lineal, Ramdom Forest y SVR, obviamente todos enfocados hacia un problema de regresión. Finalmente con todos los modelos realizaremos una evaluación de los resultados obtenidos a fin de sustentar una conclusión final para este proyecto.

1 1. Importación del Dataset

Primero importamos el conjunto de datos mediante read_csv de pandas y observamos su organización.

[2]:
```
data = pd.read_csv('airbnb-listings-extract.csv', sep=';',␣
↪error_bad_lines=False)
```

```
C:\Users\lolo\AppData\Local\Temp/ipykernel_2416/828998556.py:1: FutureWarning:
The error_bad_lines argument has been deprecated and will be removed in a future
version. Use on_bad_lines in the future.


  data = pd.read_csv('airbnb-listings-extract.csv', sep=';',
error_bad_lines=False)
```

[3]:
```
data.describe()
```

[3]:

	ID	Scrape ID	Host ID	Host Response Rate	\
count	1.478000e+04	1.478000e+04	1.478000e+04	12881.000000	
mean	1.028089e+07	2.017037e+13	3.608080e+07	94.823461	
std	5.564829e+06	5.667971e+08	3.425360e+07	15.215988	
min	1.862800e+04	2.016010e+13	1.745300e+04	0.000000	
25%	5.554732e+06	2.017041e+13	6.787360e+06	100.000000	
50%	1.133492e+07	2.017041e+13	2.464875e+07	100.000000	
75%	1.532631e+07	2.017041e+13	5.432919e+07	100.000000	
max	1.910969e+07	2.017062e+13	1.247534e+08	100.000000	

	Host Listings Count	Host Total Listings Count	Latitude	\
count	14777.000000	14777.000000	14780.000000	
mean	12.513636	12.513636	40.497626	
std	34.090223	34.090223	4.641387	
min	0.000000	0.000000	-37.851182	
25%	1.000000	1.000000	40.409726	
50%	2.000000	2.000000	40.419466	
75%	6.000000	6.000000	40.430916	
max	519.000000	519.000000	55.966912	

	Longitude	Accommodates	Bathrooms	…	Number of Reviews	\
count	14780.000000	14780.000000	14725.000000	…	14780.000000	
mean	-3.858041	3.277808	1.281732	…	22.632273	
std	14.123146	2.097291	0.658517	…	38.290244	
min	-123.131344	1.000000	0.000000	…	0.000000	
25%	-3.707604	2.000000	1.000000	…	1.000000	
50%	-3.700785	3.000000	1.000000	…	7.000000	
75%	-3.684057	4.000000	1.000000	…	27.000000	
max	153.371427	16.000000	8.000000	…	446.000000	

	Review Scores Rating	Review Scores Accuracy	\
count	11476.000000	11454.000000	
mean	91.697978	9.416012	
std	8.989101	0.921938	
min	20.000000	2.000000	
25%	89.000000	9.000000	
50%	94.000000	10.000000	
75%	98.000000	10.000000	
max	100.000000	10.000000	

	Review Scores Cleanliness	Review Scores Checkin	\
count	11460.000000	11443.000000	
mean	9.328883	9.621778	
std	0.989933	0.802736	
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                 11460.000000            11440.000000
mean                      9.647033                9.532168
std                       0.767116                0.774527
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         11439.000000                    14776.000000       11618.000000
mean              9.218201                        9.486871           1.870014
std               0.950578                       23.626014           1.867550
min               2.000000                        1.000000           0.020000
25%               9.000000                        1.000000           0.450000
50%               9.000000                        2.000000           1.200000
75%              10.000000                        5.000000           2.780000
max              10.000000                      145.000000          17.210000

[8 rows x 36 columns]
```

[4]:
```
data.head()
```

[4]:
```
          ID                             Listing Url       Scrape ID  \
0   11210388  https://www.airbnb.com/rooms/11210388  20170306202425
1   17471131  https://www.airbnb.com/rooms/17471131  20170407214050
2   17584891  https://www.airbnb.com/rooms/17584891  20170407214050
3    5398030   https://www.airbnb.com/rooms/5398030  20170407214050
4   18104606  https://www.airbnb.com/rooms/18104606  20170407214050

  Last Scraped                                  Name  \
0   2017-03-07       The Loft-Full Bath-Deck w/View
1   2017-04-08           Claris I, Friendly Rentals
2   2017-04-08  Style Terrace Red, Friendly Rentals
3   2017-04-08    Picasso Suite 1.4 Paseo de Gracia
4   2017-04-08       Smart City Centre Apartment II

                                              Summary  \
0  Loft in the Hill country 12-15 minutes directl…
1  This apartment has: 1 double bed, 1 double sof…
2  This apartment has: 1 double bed, 1 double sof…
```

```
3  Live like a local in Barcelona's most chic dis…
4  Unique apartment in vibrant neighborhoods, car…

                                                Space  \
0  This loft has a spectacular view of the hills …
1  This apartment has: 1 double bed, 1 double sof…
2  This apartment has: 1 double bed, 1 double sof…
3  You wake up to the sun rising over Barcelona's…
4  License: HUTB-005313 Charming apartment, locat…

                                           Description Experiences Offered  \
0  Loft in the Hill country 12-15 minutes directl…                 none
1  This apartment has: 1 double bed, 1 double sof…                 none
2  This apartment has: 1 double bed, 1 double sof…                 none
3  Live like a local in Barcelona's most chic dis…                 none
4  Unique apartment in vibrant neighborhoods, car…                 none

                              Neighborhood Overview  …  \
0  This neighborhood is located in the hills west…  …
1  Plaza   Cataluña Catalonia Square is the large…  …
2  Plaza   Cataluña Catalonia Square is the large…  …
3  We're right in the heart of the chic L'Eixampl…  …
4                                            NaN  …

  Review Scores Communication Review Scores Location Review Scores Value  \
0                        10.0                   10.0                10.0
1                        10.0                    8.0                10.0
2                         NaN                    NaN                 NaN
3                        10.0                   10.0                 9.0
4                         NaN                    NaN                 NaN

       License Jurisdiction Names Cancellation Policy  \
0          NaN                NaN            moderate
1  HUTB-006721                NaN     super_strict_30
2  HUTB-007527                NaN     super_strict_30
3          NaN                NaN              strict
4          NaN                NaN            flexible

  Calculated host listings count Reviews per Month  \
0                            1.0              3.50
1                          106.0              0.86
2                          106.0               NaN
3                           24.0              1.09
4                           92.0               NaN

                    Geolocation  \
0  30.3373609355,-97.8632766782
```

```
1   41.3896829422,2.17262543017
2   41.3930345489,2.16217327868
3    41.3969668101,2.1674178103
4   41.3886851936,2.15514963616
```

```
                                                Features
0  Host Is Superhost,Host Has Profile Pic,Host Id…
1  Host Has Profile Pic,Requires License,Instant …
2  Host Has Profile Pic,Requires License,Instant …
3  Host Has Profile Pic,Host Identity Verified,Re…
4  Host Has Profile Pic,Host Identity Verified,Is…

[5 rows x 89 columns]
```

El conjunto de datos tiene tal número de variables que no somos capaces de visualizarlas todas mediante estos métodos. Podemos deducir que necesitaremos enfocarnos primero en elegir y transformar variables para poder analizar con efectividad el dataset.

2 2. Transformación y eliminación de los datos

2.0.1 2.1. Selección de variables

Antes de proceder a eliminar variables y arriesgarnos a perder información importante, estudiaremos que variables son categóricas observando cuantos valores repetidos contienen y el tipo de datos en su interior.

[5]:
```
for i in data.columns:
    n = len(pd.unique(data[i]))
    print("No.of.unique values of '"+i+"':", n)
```

```
No.of.unique values of 'ID': 14780
No.of.unique values of 'Listing Url': 14780
No.of.unique values of 'Scrape ID': 38
No.of.unique values of 'Last Scraped': 37
No.of.unique values of 'Name': 14376
No.of.unique values of 'Summary': 13290
No.of.unique values of 'Space': 10175
No.of.unique values of 'Description': 14282
No.of.unique values of 'Experiences Offered': 5
No.of.unique values of 'Neighborhood Overview': 7913
No.of.unique values of 'Notes': 4653
No.of.unique values of 'Transit': 7840
No.of.unique values of 'Access': 7129
No.of.unique values of 'Interaction': 6586
No.of.unique values of 'House Rules': 8031
No.of.unique values of 'Thumbnail Url': 11958
No.of.unique values of 'Medium Url': 11958
No.of.unique values of 'Picture Url': 14758
```

```
No.of.unique values of 'XL Picture Url': 11958
No.of.unique values of 'Host ID': 8935
No.of.unique values of 'Host URL': 8935
No.of.unique values of 'Host Name': 3277
No.of.unique values of 'Host Since': 2117
No.of.unique values of 'Host Location': 569
No.of.unique values of 'Host About': 5008
No.of.unique values of 'Host Response Time': 5
No.of.unique values of 'Host Response Rate': 71
No.of.unique values of 'Host Acceptance Rate': 13
No.of.unique values of 'Host Thumbnail Url': 8894
No.of.unique values of 'Host Picture Url': 8894
No.of.unique values of 'Host Neighbourhood': 396
No.of.unique values of 'Host Listings Count': 57
No.of.unique values of 'Host Total Listings Count': 57
No.of.unique values of 'Host Verifications': 207
No.of.unique values of 'Street': 1646
No.of.unique values of 'Neighbourhood': 391
No.of.unique values of 'Neighbourhood Cleansed': 478
No.of.unique values of 'Neighbourhood Group Cleansed': 50
No.of.unique values of 'City': 247
No.of.unique values of 'State': 103
No.of.unique values of 'Zipcode': 620
No.of.unique values of 'Market': 44
No.of.unique values of 'Smart Location': 257
No.of.unique values of 'Country Code': 18
No.of.unique values of 'Country': 18
No.of.unique values of 'Latitude': 14780
No.of.unique values of 'Longitude': 14780
No.of.unique values of 'Property Type': 22
No.of.unique values of 'Room Type': 3
No.of.unique values of 'Accommodates': 16
No.of.unique values of 'Bathrooms': 18
No.of.unique values of 'Bedrooms': 11
No.of.unique values of 'Beds': 17
No.of.unique values of 'Bed Type': 5
No.of.unique values of 'Amenities': 12380
No.of.unique values of 'Square Feet': 104
No.of.unique values of 'Price': 371
No.of.unique values of 'Weekly Price': 389
No.of.unique values of 'Monthly Price': 587
No.of.unique values of 'Security Deposit': 125
No.of.unique values of 'Cleaning Fee': 112
No.of.unique values of 'Guests Included': 15
No.of.unique values of 'Extra People': 68
No.of.unique values of 'Minimum Nights': 43
No.of.unique values of 'Maximum Nights': 173
No.of.unique values of 'Calendar Updated': 57
```

```
No.of.unique values of 'Has Availability': 2
No.of.unique values of 'Availability 30': 31
No.of.unique values of 'Availability 60': 61
No.of.unique values of 'Availability 90': 91
No.of.unique values of 'Availability 365': 366
No.of.unique values of 'Calendar last Scraped': 38
No.of.unique values of 'Number of Reviews': 263
No.of.unique values of 'First Review': 1680
No.of.unique values of 'Last Review': 786
No.of.unique values of 'Review Scores Rating': 55
No.of.unique values of 'Review Scores Accuracy': 10
No.of.unique values of 'Review Scores Cleanliness': 10
No.of.unique values of 'Review Scores Checkin': 10
No.of.unique values of 'Review Scores Communication': 9
No.of.unique values of 'Review Scores Location': 9
No.of.unique values of 'Review Scores Value': 10
No.of.unique values of 'License': 317
No.of.unique values of 'Jurisdiction Names': 13
No.of.unique values of 'Cancellation Policy': 8
No.of.unique values of 'Calculated host listings count': 46
No.of.unique values of 'Reviews per Month': 842
No.of.unique values of 'Geolocation': 14780
No.of.unique values of 'Features': 89
```

[6]:
```
data.dtypes
```

[6]:
```
ID                                 int64
Listing Url                       object
Scrape ID                          int64
Last Scraped                      object
Name                              object
                                  …
Cancellation Policy               object
Calculated host listings count    float64
Reviews per Month                 float64
Geolocation                       object
Features                          object
Length: 89, dtype: object
```

Tal y como ya hemos fijado previamente, la variable regresora objetivo que queremos obtener será el precio, “Price”.

[7]:
```
y = data['Price']
```

A continuación escogemos las variables con las que vamos a trabajar a la hora de crear el evaluador. Algunas variables las eliminamos por ser muy parecidas entre si y por tanto, redundantes, otras por no aportar información del problema (Como, por ejemplo, números de identificación) y otras como ‘Monthly Price’ por ser hasta cierto punto derivadas de nuestra variable objetivo.

[8]:
```
data=data[['Host Since','Host Response Rate','Host Acceptance Rate',
           'Host Listings Count','Host Total Listings Count',
           'Host Verifications','Market','Property Type','Room Type',
           'Accommodates','Bathrooms','Bedrooms','Beds','Bed Type',
           'Amenities','Square Feet',
           'Cleaning Fee','Guests Included','Extra People','Minimum Nights',
           'Maximum Nights','Availability 365','Number of Reviews',
           'Review Scores Rating','License','Cancellation Policy',
            'Calculated host listings count','Reviews per Month','Geolocation',
            'Features','Latitude','Longitude']
          ]
```

[9]:
```
data.shape
```

[9]:
```
(14780, 32)
```

[10]:
```
data.head()
```

[10]:

	Host Since	Host Response Rate	Host Acceptance Rate	Host Listings Count	\
0	2016-02-11	100.0	NaN	1.0	
1	2016-11-08	100.0	NaN	162.0	
2	2016-11-08	100.0	NaN	162.0	
3	2010-05-25	100.0	NaN	27.0	
4	2016-01-12	99.0	NaN	207.0	

	Host Total Listings Count	\
0	1.0	
1	162.0	
2	162.0	
3	27.0	
4	207.0	

	Host Verifications	Market	Property Type	\
0	email,phone,reviews,kba	Austin	Loft	
1	email,phone,reviews,work_email	Barcelona	Apartment	
2	email,phone,reviews,work_email	Barcelona	Apartment	
3	email,phone,facebook,reviews,manual_offline,ju…	Barcelona	Apartment	
4	email,phone,reviews,jumio	Barcelona	Apartment	

	Room Type	Accommodates	…	Number of Reviews	\
0	Entire home/apt	2	…	42	
1	Entire home/apt	4	…	1	
2	Entire home/apt	4	…	0	
3	Entire home/apt	8	…	13	
4	Entire home/apt	5	…	0	

	Review Scores Rating	License	Cancellation Policy	\

```
0                  98.0          NaN            moderate
1                  80.0  HUTB-006721     super_strict_30
2                   NaN  HUTB-007527     super_strict_30
3                  92.0          NaN              strict
4                   NaN          NaN            flexible

  Calculated host listings count  Reviews per Month  \
0                            1.0               3.50
1                          106.0               0.86
2                          106.0                NaN
3                           24.0               1.09
4                           92.0                NaN

                    Geolocation  \
0  30.3373609355,-97.8632766782
1   41.3896829422,2.17262543017
2   41.3930345489,2.16217327868
3    41.3969668101,2.1674178103
4   41.3886851936,2.15514963616

                                            Features   Latitude  Longitude
0  Host Is Superhost,Host Has Profile Pic,Host Id…  30.337361 -97.863277
1  Host Has Profile Pic,Requires License,Instant …  41.389683   2.172625
2  Host Has Profile Pic,Requires License,Instant …  41.393035   2.162173
3  Host Has Profile Pic,Host Identity Verified,Re…  41.396967   2.167418
4  Host Has Profile Pic,Host Identity Verified,Is…  41.388685   2.155150

[5 rows x 32 columns]
```

Ahora los datos tienen una organización fácil de visualizar, sin variables redundantes o ajenas a nuestros intereses.

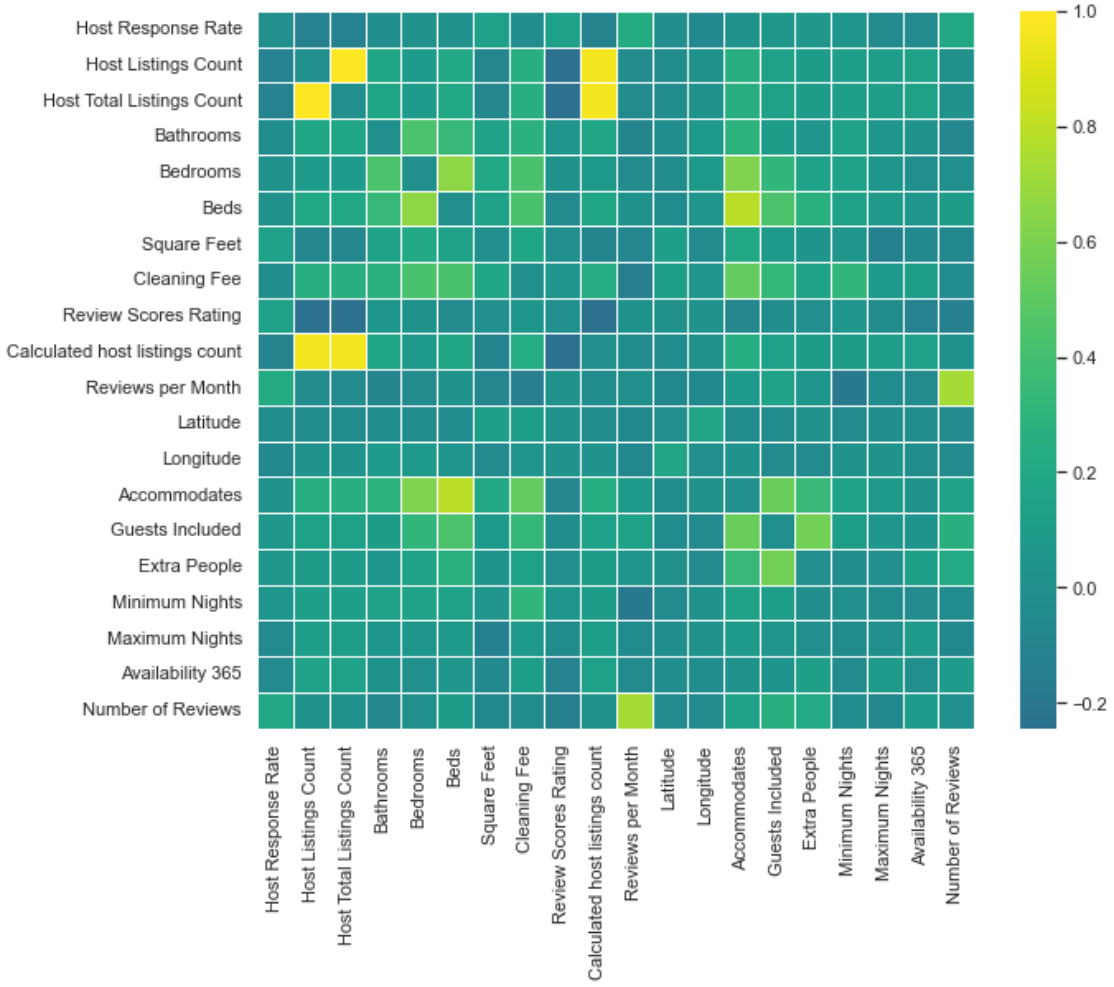
2.0.2 2.2. Matriz de correlación en variables continuas

Construimos y estudiamos la matriz de correlación para averiguar si algunas variables aportan información demasiado similar entre sí mismas a nuestro modelo de regresión, lo que permitiría eliminar algunas variables para simplificar los datos.

[11]:
```
list_var_continuous = list(data.select_dtypes('float').columns)+list(data.
↪select_dtypes('int').columns)
```

[12]:
```
import funciones_auxiliares
```

[13]:
```
funciones_auxiliares.get_corr_matrix(dataset = data[list_var_continuous],␣
↪metodo='spearman', size_figure=[10,8])
```



[13]: 0

Con un simple análisis cualitativo parece claro que 3 de las variables relacionadas tienen el máximo grado de correlación posible por lo que podemos prescindir de dos de ellas para así, simplificar levemente el modelo.

[14]:
```
data = data.drop(['Host Total Listings Count', 'Host Listings Count'], axis = 1)
```

[15]:
```
data.columns
```

[15]:
```
Index(['Host Since', 'Host Response Rate', 'Host Acceptance Rate',
       'Host Verifications', 'Market', 'Property Type', 'Room Type',
       'Accommodates', 'Bathrooms', 'Bedrooms', 'Beds', 'Bed Type',
       'Amenities', 'Square Feet', 'Cleaning Fee', 'Guests Included',
       'Extra People', 'Minimum Nights', 'Maximum Nights', 'Availability 365',
       'Number of Reviews', 'Review Scores Rating', 'License',
       'Cancellation Policy', 'Calculated host listings count',
```

```
       'Reviews per Month', 'Geolocation', 'Features', 'Latitude',
       'Longitude'],
      dtype='object')
```

3 3.Transformación de variables

3.0.1 3.1. Cancellation Policy

Utilizaremos los cuantiles para poder distibuir los valores de la variable "Cancellation Policy" en cuatro categorías y así la transformaremos en una variable con la que podremos trabajar eficientemente.

[16]:
```
data['Cancellation Policy'].value_counts()
```

[16]:
```
strict             5742
flexible           4671
moderate           4248
strict_new           34
super_strict_60      32
moderate_new         20
super_strict_30      19
flexible_new         14
Name: Cancellation Policy, dtype: int64
```

[17]:
```
data['Cancellation Policy'] = data['Cancellation Policy'].replace({'strict': 0.
↪25, 'flexible':0.9, 'moderate': 0.5,
                                                                   'strict_new':
↪ 0.25, 'super_strict_60':0, 'super_strict_30':0,
                                                                              ␣
↪'moderate_new':0.5, 'flexible_new': 0.9}).astype('float')
```

Al estandarizar los datos con la división del X_train y X_test se conseguirá mejorar el algoritmo

3.0.2 3.3. Host Since

El siguiente paso será transformar la fecha desde la que el dueño del airbnb ofrece un alojamiento "Host Since" en días desde esa fecha.

[18]:
```
fecha = pd.to_datetime(data['Host Since'])
#Creamos una variable que reste la maxima fecha de nuestra base de datos a la␣
↪fecha del anfitrión
tiempo_anfitrión = (max(fecha) - fecha).astype('timedelta64[D]')
#Sustituimos por la variable fecha para que sea más sencillo utilizarla a la␣
↪hora de los modelos
data['Host Since'] = tiempo_anfitrión
data['Host Since'] = data['Host Since'].fillna(0)
```

3.0.3 3.4. Amenities, Features, Host Verifications y License

En esta sección trataremos las variables que estaban en formato string y las transformaremos en ristras de variables binarias. Posteriormente usaremos análisis de componentes principales (PCA) para reducirlas.

[19]:
```
def str_to_dataframe(data,column):
    datos=[]
    for i in data[column]:
        if type(i)== float:#si es np.nan creamos un diccionario vacio
            lista={}
        else:#en caso contrario creamos un diccionario con los datos
            aux =i.split(',')
            lista={}
            for j in aux:
                lista[j]=True
        datos.append(lista)#añadimos el diccionario creada a la lista
    return(pd.DataFrame(datos).fillna(False))#transformamos la list en␣
 ↪dataframe
```

Dado que todas las variables que nos saldrán son dicotómicas usaremos el PCA para tranformarlas en variables continuas. (Previamente probamos usar clustering jerárquico sobre estas variables tomando la distancia de Manhattan pero dado que el tiempo de carga era excesivo y los resultados no presentaban mejores cualidades que los obtenidos por PCA, descartamos el método).

[20]:
```
def str_to_continuous(data, columns,N_components):
    '''
    El siguiente código
    data: dataframe a usar
    columns: columnar que se quieren transformar
    n_components:lista con numero de componentes que se quieren obtener por␣
 ↪fila

    '''
    from sklearn.decomposition import PCA
    X_transK=pd.DataFrame()
    for i,column in enumerate(columns):
        datos=str_to_dataframe(data,column)
        if column=='Host Verifications':#Contaremos License como parte de host␣
 ↪verification
            datos['License']=data['License'].isna()
        pca = PCA(n_components=N_components[i])
        pca.fit(datos)
        print(pca.explained_variance_ratio_.sum())
        labels=[]
        for i in range(N_components[i]):
            labels.append(column+'_'+str(i))
```

```
        data= data.join(pd.DataFrame(pca.transform(datos),columns=labels))
    return data
```

Tras varias pruebas decidimos utilizar 12 componentes para “Amenities”, 3 para “Features” y 3 para “Host Verifications”.

[21]:
```
columns=['Amenities','Features','Host Verifications']
N_components=[12,3,3]
data=str_to_continuous(data, columns,N_components)
```

```
0.62903370170837
0.7710154876416929
0.6157291122889633
```

[22]:
```
data=data.drop(['Amenities','Features','Host Verifications','License'],axis=1)
```

[23]:
```
data
```

[23]:

	Host Since	Host Response Rate	Host Acceptance Rate	Market	\
0	421.0	100.0	NaN	Austin	
1	150.0	100.0	NaN	Barcelona	
2	150.0	100.0	NaN	Barcelona	
3	2509.0	100.0	NaN	Barcelona	
4	451.0	99.0	NaN	Barcelona	
…	…	…	…	…	
14775	1206.0	100.0	NaN	New York	
14776	14.0	NaN	NaN	Paris	
14777	641.0	100.0	NaN	Paris	
14778	1479.0	100.0	80%	Denver	
14779	1082.0	100.0	100%	Seattle	

	Property Type	Room Type	Accommodates	Bathrooms	Bedrooms	\
0	Loft	Entire home/apt	2	1.0	1.0	
1	Apartment	Entire home/apt	4	1.0	1.0	
2	Apartment	Entire home/apt	4	1.0	1.0	
3	Apartment	Entire home/apt	8	2.0	3.0	
4	Apartment	Entire home/apt	5	1.0	2.0	
…	…	…	…	…	…	
14775	Apartment	Entire home/apt	2	1.0	1.0	
14776	Bed & Breakfast	Private room	1	1.0	1.0	
14777	Apartment	Entire home/apt	5	1.0	2.0	
14778	Apartment	Entire home/apt	4	1.0	1.0	
14779	House	Private room	2	1.0	1.0	

	Beds	…	Amenities_8	Amenities_9	Amenities_10	Amenities_11	\
0	1.0	…	-0.456836	0.292290	0.606427	0.123177	
1	1.0	…	0.330704	0.092580	0.809762	0.774670	

```
2       1.0  …    0.330704     0.092580      0.809762      0.774670
3       4.0  …   -0.697000     0.592590      0.065435     -0.112765
4       2.0  …    0.026428     0.186452     -0.049936     -0.078339
…     …  …          …           …            …             …
14775   2.0  …    0.318200     0.250950      0.405232      0.051795
14776   2.0  …   -0.046563    -0.329755      0.337282     -0.310132
14777   3.0  …   -0.196359    -0.390201     -0.253424      0.824589
14778   2.0  …   -0.426714    -0.522482     -0.303882      0.970929
14779   1.0  …   -0.537179    -0.532731     -0.623704      0.642655

       Features_0  Features_1  Features_2  Host Verifications_0  \
0       -0.441616   -0.630208   -0.066578             -0.619940
1        0.531915    0.912232    0.267454             -0.541394
2        0.531915    0.912232    0.267454             -0.541394
3       -0.258650    0.582044    0.767222              0.408022
4       -0.717360    0.388629   -0.098217              0.218679
…           …          …          …                      …
14775   -0.441616   -0.630208   -0.066578              0.218679
14776    0.053822    0.666518   -0.601453             -0.711725
14777    0.017094   -0.436794    0.798862              0.218679
14778   -0.347748   -0.583844   -0.100140              0.386172
14779   -0.736743    0.336329   -0.101685             -0.332534

       Host Verifications_1  Host Verifications_2
0                 -0.101797              0.106469
1                 -0.128328             -0.093625
2                 -0.128328             -0.093625
3                  0.634086             -0.754867
4                 -0.313825             -0.301971
…                     …                    …
14775             -0.313825             -0.301971
14776              0.004053              0.461924
14777             -0.313825             -0.301971
14778              0.660100             -0.476575
14779              0.881767              0.143480

[14780 rows x 44 columns]
```

3.0.4 3.5. Property Type

A la hora de estudiar la variable "Property Type" observamos que la gran mayoría de alojamientos en el dataset son apartamentos, casas y construcciones parecidas y una minoría de alojamientos pertenece a una gran cantidad de diferentes tipos de vivienda bastante inusuales en el dataset, como barcos o bungalows.

Por tanto englobaremos todos los alojamientos en tres categorías, apartamentos, casas u otros tipos de vivienda, para así seguir simplificando los datos.

[24]:
```
data['Property Type'].value_counts()
```

[24]:
```
Apartment             11962
House                  1368
Condominium             363
Bed & Breakfast         353
Loft                    305
Other                   225
Dorm                     44
Guesthouse               42
Chalet                   26
Villa                    19
Townhouse                17
Hostel                   16
Serviced apartment       13
Boutique hotel            6
Boat                      4
Camper/RV                 4
Guest suite               3
Casa particular           3
Earth House               3
Bungalow                  2
Tent                      1
Timeshare                 1
Name: Property Type, dtype: int64
```

[25]:
```
data['Property Type'].replace({
    'Townhouse': 'House',
    'Serviced apartment': 'Apartment',
    'Loft': 'Apartment',
    'Bungalow': 'House',
    'Guesthouse': 'House',
    'Casa particular': 'House',
    'Cottage': 'House',
    'Villa': 'House',
    'Tiny house': 'House',
    'Earth House': 'House',
    'Chalet': 'House',
    'Boutique hotel': 'Other',
    'Condominium': 'Other',
    'Bed & Breakfast': 'Other',
    'Dorm': 'Other',
    'Hostel': 'Other',
    'Guest suite': 'Other',
    'Tent': 'Other',
    'Timeshare': 'Other',
    'Boat': 'Other',
```

```
    'Camper/RV': 'Other'
    }, inplace=True)
```

[26]:
```
data['Property Type'].value_counts()
```

[26]:
```
Apartment    12280
House         1480
Other         1020
Name: Property Type, dtype: int64
```

3.0.5 3.6. Geolocalización

Dado que la variable "Geolocation" tiene dos componentes, latitud y longitud, crearemos una nueva variable que estará constituida por el producto entre la latitud y longitud de las coordenadas en las que estaba el alojamiento, de tal forma reducimos las variables. A parte las viviendas cercanas entre sí tendrán valores similares lo cual ofrece información más valiosa que la latitud y la longitud por separado.

[27]:
```
data['Geolocation']=data['Latitude']*data['Longitude']
```

[28]:
```
data=data.drop(['Longitude','Latitude'],1)
```

```
C:\Users\lolo\AppData\Local\Temp/ipykernel_2416/3581283567.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only.
  data=data.drop(['Longitude','Latitude'],1)
```

3.0.6 3.7 Host acceptance rate

En esta última transformación trataremos la variable "Host Acceptance Rate". En este caso, el método usado para representar el porcentaje no es práctico a la hora de usar los modelos que tenemos preparados, así que simplemente cambiaremos el formato de la variable para evitar inconvenientes.

[29]:
```
lista=[]
for i in data['Host Acceptance Rate']:
    lista.append(float(str(i).replace('%','')))
data['Host Acceptance Rate']=lista
```

4 4. Imputación de Valores Missing

Los conjuntos de datos pueden tener valores faltantes y esto puede causar problemas para muchos algoritmos de aprendizaje automático. Por ejemplo, el algoritmo Ramdom Forest no puede trabajar con un dataset con datos faltantes.

Por tanto, ahora procederemos a identificar y reemplazar los valores perdidos para cada columna en sus datos de entrada antes de modelar su tarea de predicción, es decir, imputaremos los datos perdidos. Primero nos centraremos en la variables continuas para proceder después con las categóricas.

4.0.1 4.1. Variables Continuas

[30]:
```
list_var_continuous = list(data.select_dtypes('float').columns)+list(data.
↪select_dtypes('int').columns)
list_var_discrets = list(data.select_dtypes('object').columns)
```

[31]:
```
data[list_var_continuous]
```

[31]:

	Host Since	Host Response Rate	Host Acceptance Rate	Bathrooms	\
0	421.0	100.0	NaN	1.0	
1	150.0	100.0	NaN	1.0	
2	150.0	100.0	NaN	1.0	
3	2509.0	100.0	NaN	2.0	
4	451.0	99.0	NaN	1.0	
…	…	…	…	…	
14775	1206.0	100.0	NaN	1.0	
14776	14.0	NaN	NaN	1.0	
14777	641.0	100.0	NaN	1.0	
14778	1479.0	100.0	80.0	1.0	
14779	1082.0	100.0	100.0	1.0	

	Bedrooms	Beds	Square Feet	Cleaning Fee	Review Scores Rating	\
0	1.0	1.0	NaN	NaN	98.0	
1	1.0	1.0	NaN	40.0	80.0	
2	1.0	1.0	NaN	60.0	NaN	
3	3.0	4.0	NaN	79.0	92.0	
4	2.0	2.0	NaN	55.0	NaN	
…	…	…	…	…	…	
14775	1.0	2.0	NaN	60.0	95.0	
14776	1.0	2.0	NaN	NaN	NaN	
14777	2.0	3.0	NaN	40.0	94.0	
14778	1.0	2.0	NaN	10.0	100.0	
14779	1.0	1.0	NaN	NaN	98.0	

	Cancellation Policy	…	Host Verifications_0	Host Verifications_1	\
0	0.50	…	-0.619940	-0.101797	
1	0.00	…	-0.541394	-0.128328	
2	0.00	…	-0.541394	-0.128328	
3	0.25	…	0.408022	0.634086	
4	0.90	…	0.218679	-0.313825	
…	…	…	…	…	
14775	0.25	…	0.218679	-0.313825	
14776	0.90	…	-0.711725	0.004053	
14777	0.25	…	0.218679	-0.313825	
14778	0.50	…	0.386172	0.660100	
14779	0.90	…	-0.332534	0.881767	

```
       Host Verifications_2  Accommodates  Guests Included  Extra People  \
0                  0.106469             2                1             0
1                 -0.093625             4                2            44
2                 -0.093625             4                2            42
3                 -0.754867             8                2            39
4                 -0.301971             5                1             0
…                       …             …                …             …
14775             -0.301971             2                1             0
14776              0.461924             1                1             0
14777             -0.301971             5                3            10
14778             -0.476575             4                2            10
14779              0.143480             2                2             0

       Minimum Nights  Maximum Nights  Availability 365  Number of Reviews
0                   2            1125               281                 42
1                   1            1125               309                  1
2                   1            1125               291                  0
3                   2            1125               316                 13
4                   2            1125                 7                  0
…                   …               …                 …                  …
14775               1            1125               339                 30
14776               3               7                78                  0
14777               3            1125                97                 17
14778               2               7                14                  1
14779               2            1125               365                  8

[14780 rows x 38 columns]
```

[32]:
```
lista=data[list_var_continuous].isna().sum()
var_na=[]
for i,j in enumerate(lista):
    if lista[list_var_continuous[i]]>0:
        var_na.append(list_var_continuous[i])
```

[33]:
```
data[var_na].isna().mean()
```

[33]:
```
Host Response Rate                0.128484
Host Acceptance Rate              0.997361
Bathrooms                         0.003721
Bedrooms                          0.001691
Beds                              0.003315
Square Feet                       0.959540
Cleaning Fee                      0.412246
Review Scores Rating              0.223545
Calculated host listings count    0.000271
Reviews per Month                 0.213938
dtype: float64
```

En la variable “Cleaning Fee” observamos que la falta de valor significa que dicho alojamiento no cobra tasa de limpieza. Por tanto sustituimos los valores faltantes por la cantidad 0.

[34]:
```
data['Cleaning Fee']=data['Cleaning Fee'].fillna(0)
```

Las variables que hacen referencia al número de recursos que hay en un alojamiento, “Bathrooms”, “Bedrooms” y “Beds” presentan una situación parecida. Por tanto sustituimos los valores faltantes por la cantidad 0, para representar la falta de baños, dormitorios o camas en un alojamiento.

[35]:
```
data['Bathrooms']=data['Bathrooms'].fillna(0)
data['Bedrooms']=data['Bedrooms'].fillna(0)
data['Beds']=data['Beds'].fillna(0)
```

Hay dos variables que trataban información que podría haber sido muy útil al proyecto, “Host Acceptance Rate” y “Square Feet”, sin embargo ambas están compuestas en mas de un 96% de datos faltantes y por tanto carecen de utilidad a la hora de crear el evaluador. En conclusión eliminamos ambas variables.

[36]:
```
data=data.drop(['Host Acceptance Rate','Square Feet'],1)
```

```
C:\Users\lolo\AppData\Local\Temp/ipykernel_2416/1831766833.py:1: FutureWarning:
In a future version of pandas all arguments of DataFrame.drop except for the
argument 'labels' will be keyword-only.
  data=data.drop(['Host Acceptance Rate','Square Feet'],1)
```

En el caso de las variables “Host Response Rate”, “Review Scores Rating”, “Calculated host listings count” y “Reviews per Month”, los datos faltantes hacen referencia a usuarios que llevan tan poco tiempo usando la plataforma que todavía no han podido participar en ninguna de las operaciones registradas por estas variables. Consecuentemente, estableceremos que los datos faltantes sean 0.

[37]:
```
data['Host Response Rate']=data['Host Response Rate'].fillna(0)
data['Review Scores Rating']=data['Review Scores Rating'].fillna(0)
data['Calculated host listings count']=data['Calculated host listings count'].
 ↪fillna(0)
data['Reviews per Month']=data['Reviews per Month'].fillna(0)
```

Finalmente para el resto de datos faltantes en variables continuas aplicamos un imputador simple que use la media como estrategia.

[38]:
```
list_var_continuous = list(data.select_dtypes('float').columns)

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(data[list_var_continuous])
data[list_var_continuous]=imputer.transform(data[list_var_continuous])
```

4.0.2 4.2. Variables Categóricas

[39]:
```
list_var_discrets = list(data.select_dtypes('object').columns)
list_var_discrets
```

[39]:
```
['Market', 'Property Type', 'Room Type', 'Bed Type']
```

[40]:
```
data[list_var_discrets].isna().mean()
```

[40]:
```
Market           0.003857
Property Type    0.000000
Room Type        0.000000
Bed Type         0.000000
dtype: float64
```

[41]:
```
data[list_var_discrets]
```

[41]:

	Market	Property Type	Room Type	Bed Type
0	Austin	Apartment	Entire home/apt	Real Bed
1	Barcelona	Apartment	Entire home/apt	Real Bed
2	Barcelona	Apartment	Entire home/apt	Real Bed
3	Barcelona	Apartment	Entire home/apt	Real Bed
4	Barcelona	Apartment	Entire home/apt	Real Bed
…	…	…	…	…
14775	New York	Apartment	Entire home/apt	Real Bed
14776	Paris	Other	Private room	Real Bed
14777	Paris	Apartment	Entire home/apt	Real Bed
14778	Denver	Apartment	Entire home/apt	Real Bed
14779	Seattle	House	Private room	Real Bed

[14780 rows x 4 columns]

Las variables categóricas nos presentan muchos menos inconvenientes que las continuas, siendo "Market" la única de ellas con valores faltantes. Tras una leve investigación descubrimos que esta variable designa en que zona se situa el alojamiento, como el sistema ofrece zonas determinadas deducimos que los datos faltantes representan que el alojamiento está en zonas no registradas en el sistema a las que llamaremos "Other".

[42]:
```
data['Market']=data['Market'].fillna('Other')
```

5 5. Estudio de la Variable Objetivo

Siendo "Price" nuestra variable objetivo la hemos dejado apartada durante las fases anteriores del preprocesado de datos. A continuación estudiaremos esta variable y sus propiedades en el dataset.

5.0.1 5.1. Distribución de los Alojamientos según la Variable Objetivo (Búsqueda de Outliers)

[43]:
```
plt.figure(figsize=(20,4))
y.hist(bins=100, range=(0,max(y)))
plt.margins(x=0)
plt.axvline(200, color='red', linestyle='--')
plt.title("Número de Airbnb por precio", fontsize=16)
plt.xlabel("Precio ")
plt.ylabel("Número de Airbnb")
plt.show()
```

En vista de este gráfico y debido a la baja frecuencia relativa de algunos precios, vamos a filtrar el dataset. Escogeremos las observaciones correspondientes a los alojamientos con precios menores a 220.

[44]:
```
data = pd.concat([y,data],axis=1)
```

[45]:
```
datos = data.drop(data[data['Price']>220].index)
datos.shape
```

[45]: (14300, 41)

[46]:
```
datos.head()
```

[46]:

	Price	Host Since	Host Response Rate	Market	Property Type	\
0	58.0	421.0	100.0	Austin	Apartment	
1	70.0	150.0	100.0	Barcelona	Apartment	
2	70.0	150.0	100.0	Barcelona	Apartment	
4	149.0	451.0	99.0	Barcelona	Apartment	
5	106.0	451.0	99.0	Barcelona	Apartment	

	Room Type	Accommodates	Bathrooms	Bedrooms	Beds	…	Amenities_8	\
0	Entire home/apt	2	1.0	1.0	1.0	…	-0.456836	
1	Entire home/apt	4	1.0	1.0	1.0	…	0.330704	
2	Entire home/apt	4	1.0	1.0	1.0	…	0.330704	
4	Entire home/apt	5	1.0	2.0	2.0	…	0.026428	

```
5  Entire home/apt             6        1.0       2.0   3.0  …   -0.180507

   Amenities_9  Amenities_10  Amenities_11  Features_0  Features_1  \
0     0.292290      0.606427      0.123177   -0.441616   -0.630208
1     0.092580      0.809762      0.774670    0.531915    0.912232
2     0.092580      0.809762      0.774670    0.531915    0.912232
4     0.186452     -0.049936     -0.078339   -0.717360    0.388629
5     0.376710     -0.209054     -0.174186   -0.717360    0.388629

   Features_2  Host Verifications_0  Host Verifications_1  \
0   -0.066578             -0.619940             -0.101797
1    0.267454             -0.541394             -0.128328
2    0.267454             -0.541394             -0.128328
4   -0.098217              0.218679             -0.313825
5   -0.098217              0.218679             -0.313825

   Host Verifications_2
0              0.106469
1             -0.093625
2             -0.093625
4             -0.301971
5             -0.301971

[5 rows x 41 columns]
```

[47]:
```
#Actualizamos los indices
datos = datos.reset_index(drop=True)
```

Hemos reducido el dataset en unos 500 datos, y en general, los descriptivos de interés como el precio medio sigue siendo similares.

[48]:
```
datos.describe()
```

[48]:
```
              Price    Host Since  Host Response Rate  Accommodates  \
count  14283.000000  14300.000000        14300.000000  14300.000000
mean      63.710635    929.023986           82.916434      3.162657
std       40.793619    589.130465           34.467758      1.899158
min        9.000000      0.000000            0.000000      1.000000
25%       32.000000    449.750000           90.000000      2.000000
50%       55.000000    840.000000          100.000000      2.000000
75%       80.000000   1386.250000          100.000000      4.000000
max      220.000000   2882.000000          100.000000     16.000000

          Bathrooms      Bedrooms          Beds  Cleaning Fee  \
count  14300.000000  14300.000000  14300.000000  14300.000000
mean       1.235594      1.286154      1.953566     17.434685
std        0.574646      0.795692      1.445786     23.562381
```

```
min        0.000000      0.000000      0.000000      0.000000
25%        1.000000      1.000000      1.000000      0.000000
50%        1.000000      1.000000      1.000000     10.000000
75%        1.000000      2.000000      2.000000     30.000000
max        8.000000     10.000000     16.000000    662.000000

       Guests Included  Extra People  …   Amenities_8   Amenities_9  \
count     14300.000000  14300.000000  …  14300.000000  14300.000000
mean          1.554336      7.563427  …      0.000112      0.003522
std           1.050071     12.075280  …      0.416939      0.402139
min           0.000000      0.000000  …     -1.466539     -1.390883
25%           1.000000      0.000000  …     -0.280893     -0.279108
50%           1.000000      0.000000  …     -0.005002      0.005565
75%           2.000000     15.000000  …      0.273384      0.288135
max          16.000000    500.000000  …      1.617215      1.494490

       Amenities_10  Amenities_11    Features_0    Features_1    Features_2  \
count  14300.000000  14300.000000  14300.000000  14300.000000  14300.000000
mean       0.002291     -0.009623      0.000780      0.001542     -0.002481
std        0.396925      0.378928      0.509256      0.475284      0.468852
min       -1.391818     -1.179334     -0.914326     -0.689545     -0.610753
25%       -0.260908     -0.289030     -0.328365     -0.384494     -0.562877
50%        0.001217     -0.060087      0.073205     -0.201356     -0.096672
75%        0.280228      0.250938      0.462200      0.388629      0.268999
max        1.580860      1.560594      0.928753      0.914341      0.802330

       Host Verifications_0  Host Verifications_1  Host Verifications_2
count          14300.000000          14300.000000          14300.000000
mean               0.000002              0.001517              0.000338
std                0.552115              0.406380              0.357576
min               -0.783187             -0.347093             -0.793154
25%               -0.606504             -0.313825             -0.301971
50%                0.218679             -0.116631              0.108515
75%                0.348983              0.004053              0.173323
max                1.218461              1.059353              1.271921

[8 rows x 37 columns]
```

5.0.2 5.2. Imputación de Datos Missing en la Variable Objetivo

Estudiamos la variable objetivo "Price" e imputamos sus datos missing para así conseguir un dataset al que no le falten datos. Esto nos permitirá usar todos nuestros modelos sin inconvenientes.

[49]:
```
datos.isna().sum()
```

[49]:
```
Price                          17
Host Since                      0
```

```
Host Response Rate                0
Market                            0
Property Type                     0
Room Type                         0
Accommodates                      0
Bathrooms                         0
Bedrooms                          0
Beds                              0
Bed Type                          0
Cleaning Fee                      0
Guests Included                   0
Extra People                      0
Minimum Nights                    0
Maximum Nights                    0
Availability 365                  0
Number of Reviews                 0
Review Scores Rating              0
Cancellation Policy               0
Calculated host listings count    0
Reviews per Month                 0
Geolocation                       0
Amenities_0                       0
Amenities_1                       0
Amenities_2                       0
Amenities_3                       0
Amenities_4                       0
Amenities_5                       0
Amenities_6                       0
Amenities_7                       0
Amenities_8                       0
Amenities_9                       0
Amenities_10                      0
Amenities_11                      0
Features_0                        0
Features_1                        0
Features_2                        0
Host Verifications_0              0
Host Verifications_1              0
Host Verifications_2              0
dtype: int64
```

Reemplazamos los valores missing por la media.

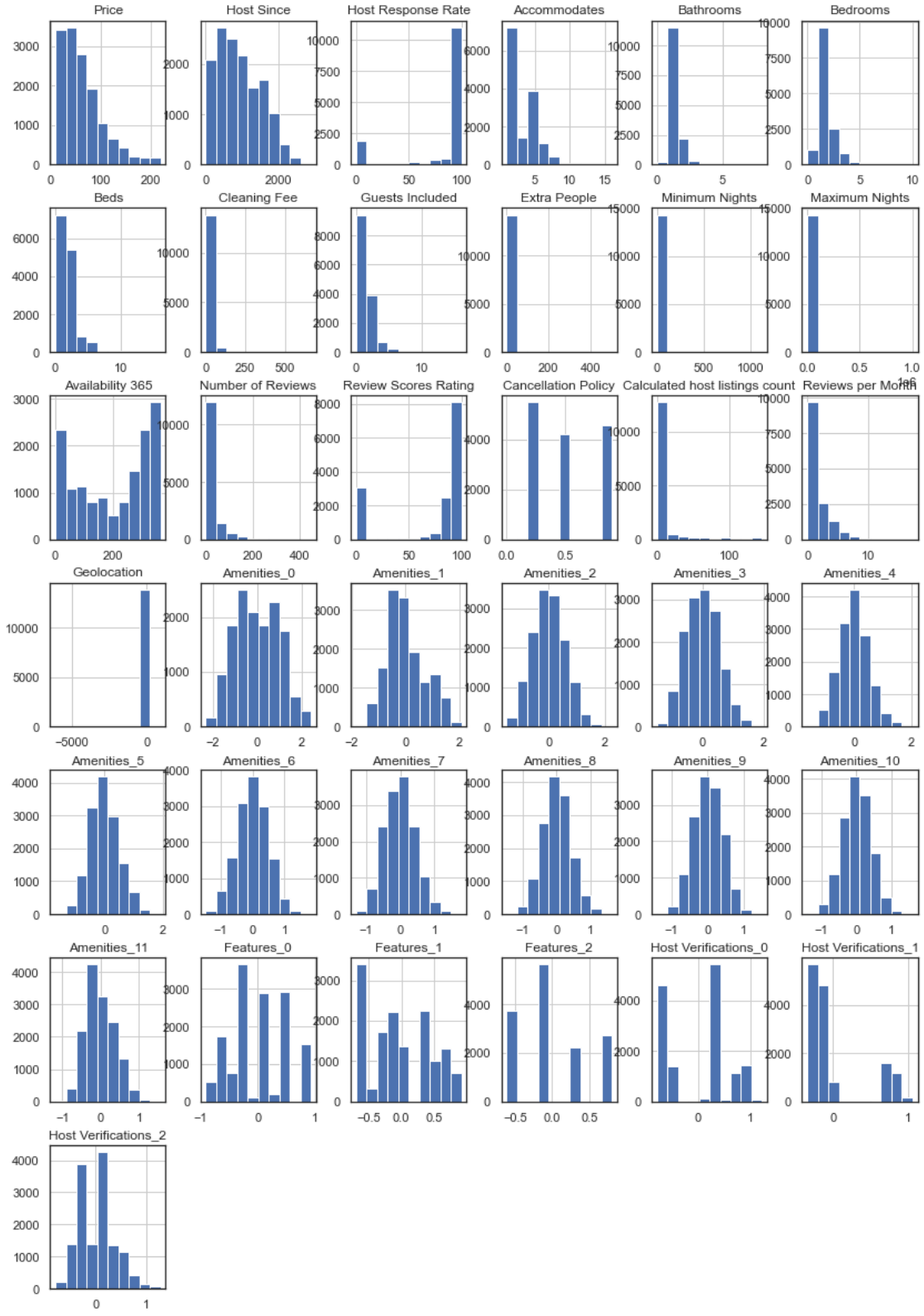
[50]:
```
datos['Price']=datos['Price'].fillna(datos['Price'].mean())
```

6 6. Gráficos de Interés

6.0.1 6.1. Histogramas de las Variables Continuas

En primer lugar, veamos los histogramas de las variables continuas para hacernos una idea de su distribución.

[51]:
```
plt.rcParams['figure.figsize'] = (14,21)
datos.hist()
plt.show()
```


Price
Host Since
Host Response Rate
Accommodates
Bathrooms
Bedrooms
Beds
Cleaning Fee
Guests Included
Extra People
Minimum Nights
Maximum Nights
Availability 365
Number of Reviews
Review Scores Rating
Cancellation Policy
Calculated host listings count
Reviews per Month
Geolocation
Amenities_0
Amenities_1
Amenities_2
Amenities_3
Amenities_4
Amenities_5
Amenities_6
Amenities_7
Amenities_8
Amenities_9
Amenities_10
Amenities_11
Features_0
Features_1
Features_2
Host Verifications_0
Host Verifications_1
Host Verifications_2

Tras los estudios realizados, vemos que las variables de "Amenities" se distribuyen aproximadamente de forma normal, aunque en general, todas las demás variables presentan ausencia de normalidad.

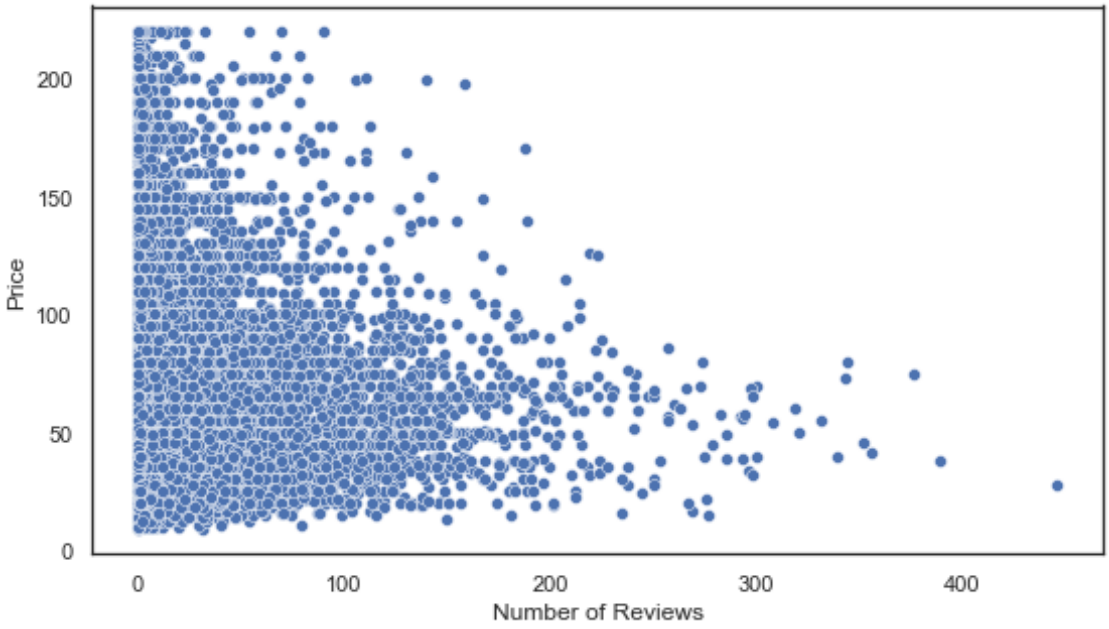
A la hora de tener en cuenta unos datos sobre otros, vemos que los valores de "Host Score Rate" suelen estar en frontera con su máximo, los valores usuales en "Accommodates" están comprendidos entre 1 y 5, observando la variables "Guests Included" deducimos que suelen haber 1 ó 2 huéspedes incluidos (Se puede deducir que uno es el anfitrión) y observando "Reviews per Month" lo más usual para los anfitriones es recibir una o dos reseñas al mes.

6.0.2 6.2. Gráficos sobre la Variable Objetivo

Vamos a estudiar gráficamente la variable objetivo "Price" en función de otras variables. De esta forma, conseguiremos hacernos una idea previa sobre que atributos podrían influir en el precio de los alojamientos.

[52]:
```
plt.rcParams['figure.figsize'] = (9,5)
sns.scatterplot(x=datos['Number of Reviews'],y=datos['Price'])
```

[52]:
```
<AxesSubplot:xlabel='Number of Reviews', ylabel='Price'>
```



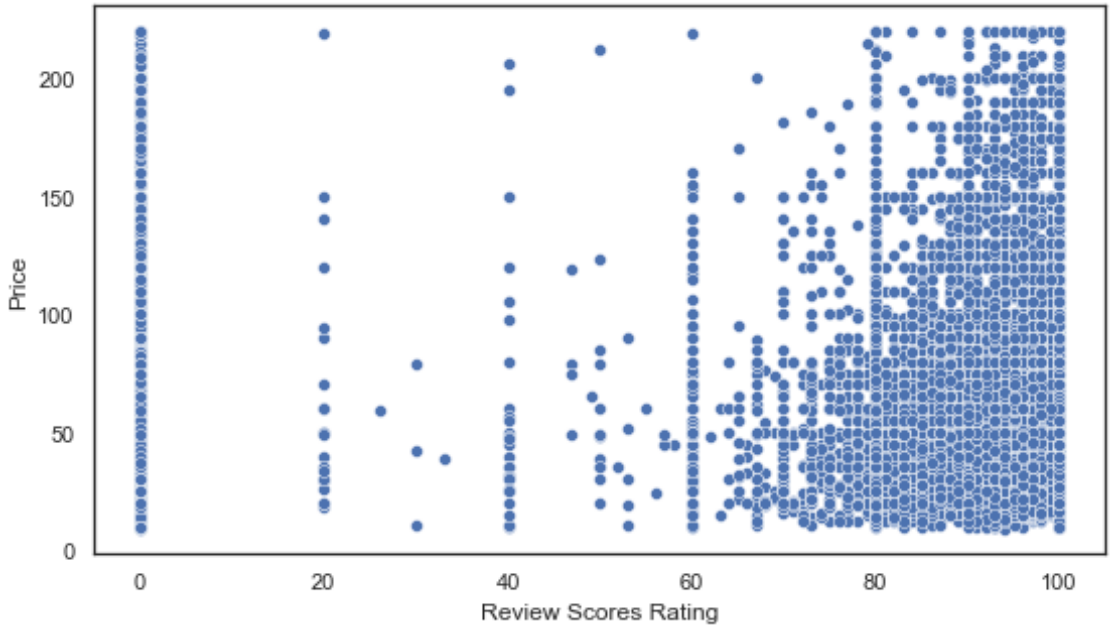
Deducimos que el número de reseñas quizás influya en el precio, ya que, salvo algún posible outlier, vemos que los alojamientos con un número elevado de reseñas suelen tener un precio bajo. De manera inversa, aquellos alojamientos con precios muy por encima de la media, tienen pocas reseñas.

Esta deducción resulta lógica. A los alojamientos con altos precios acceden menos clientes, por lo tanto, menos usuarios escriben reseñas sobre su estancia. Por el otro lado, a los alojamientos baratos accederán más clientes y estos a su vez escribirán más reseñas (En el entorno de la hostelería esto es habitual, sobretodo si el servicio ha sido una experiencia negativa, ya que los clientes usarán

las reseñas para evaluar negativamente el alojamiento).

[53]:
```
plt.rcParams['figure.figsize'] = (9,5)
sns.scatterplot(x=datos['Review Scores Rating'],y=datos['Price'])
```

[53]:
```
<AxesSubplot:xlabel='Review Scores Rating', ylabel='Price'>
```

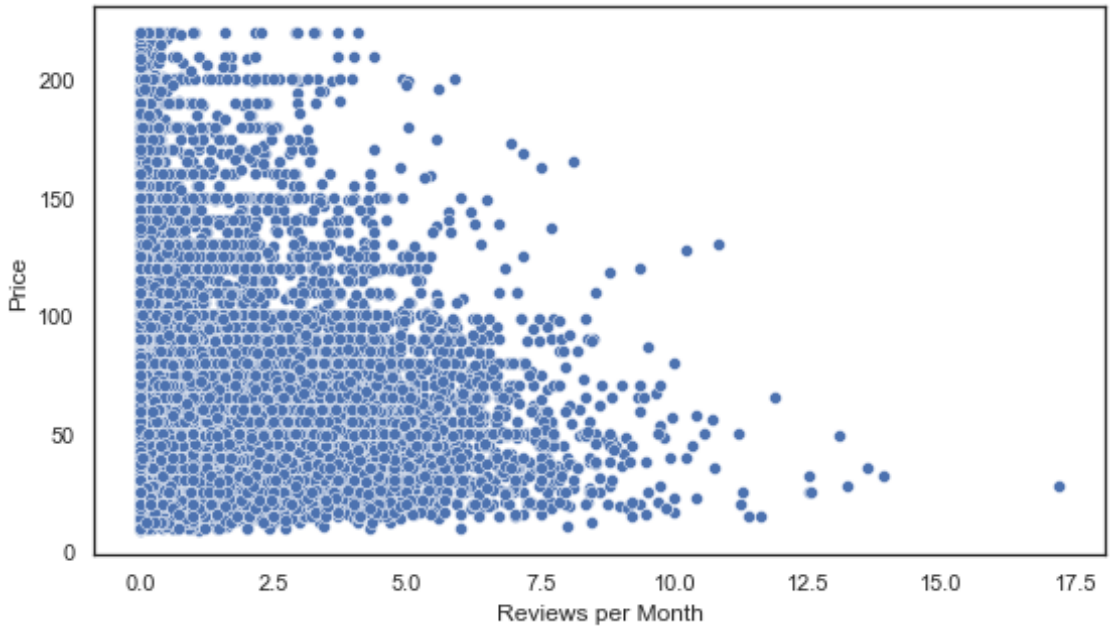


De forma similar, en consecuencia a lo que acabamos de comentar, vemos que la puntuación que ofrecen las reseñas parece relacionarse con el precio. Los alojamientos con altos precios presentan altas puntuaciones en las reseñas, mientras que los que presentan baja puntuación en las reseñas suelen ser alojamientos baratos.

No obstante, hay que resaltar que no es extraña la existencia de alojamientos baratos con muy buenas reseñas en el dataset, pero en general, los que tienen reseñas negativas son baratos.

[54]:
```
plt.rcParams['figure.figsize'] = (9,5)
sns.scatterplot(x=datos['Reviews per Month'],y=datos['Price'])
```

[54]:
```
<AxesSubplot:xlabel='Reviews per Month', ylabel='Price'>
```



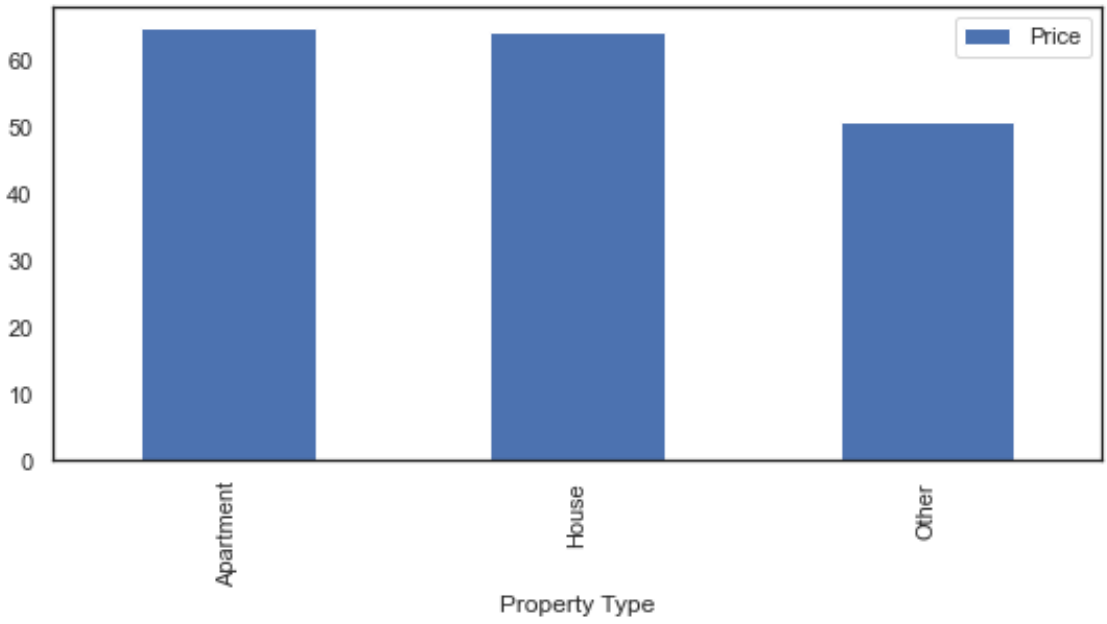
En este gráfico podemos deducir los mismos resultados que en el de "Number of Reviews". Tal vez, en el posterior análisis de regresión, se deba prescindir de esta variable.

Los gráficos de la variable objetivo en función de las demás variables continuas no arrojaban resultados de interés.

Ahora analicemos el precio medio en función de las variables categóricas:

[55]:
```
plt.rcParams['figure.figsize'] = (9,4)
pd.DataFrame(datos.groupby('Property Type')['Price'].mean().
  ↪sort_values(ascending=False)).plot(kind='bar')
```

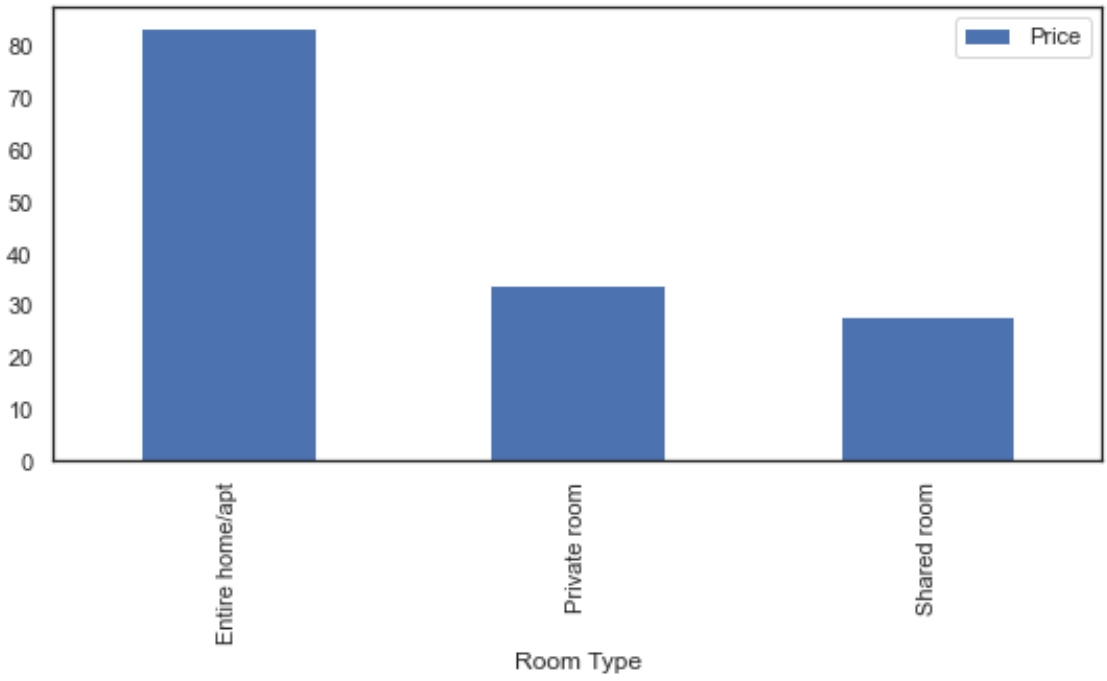
[55]:
```
<AxesSubplot:xlabel='Property Type'>
```



Podemos sospechar que no hay diferencias significativas sobre el precio entre alojarse en un apartamento y una casa. En cambio, si elegimos la otra opción el precio será inferior.

[56]:
```
plt.rcParams['figure.figsize'] = (9,4)
pd.DataFrame(datos.groupby('Room Type')['Price'].mean().
  ↪sort_values(ascending=False)).plot(kind='bar')
```

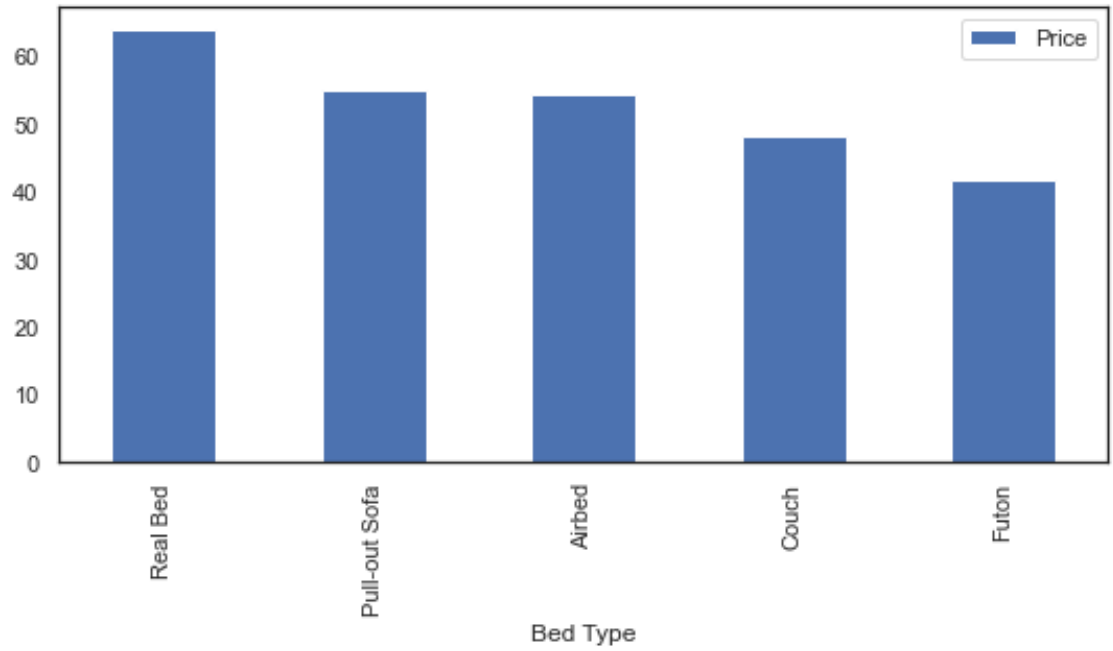
[56]:
```
<AxesSubplot:xlabel='Room Type'>
```



Era previsible imaginar que el precio de alojamiento se encarecerá si tenemos todo el inmueble a nuestra disposición. No obstante, curiosamente el precio entre una habitación privada y una compartida son similares.

[57]:
```
plt.rcParams['figure.figsize'] = (9,4)
pd.DataFrame(datos.groupby('Bed Type')['Price'].mean().
  ↪sort_values(ascending=False)).plot(kind='bar')
```

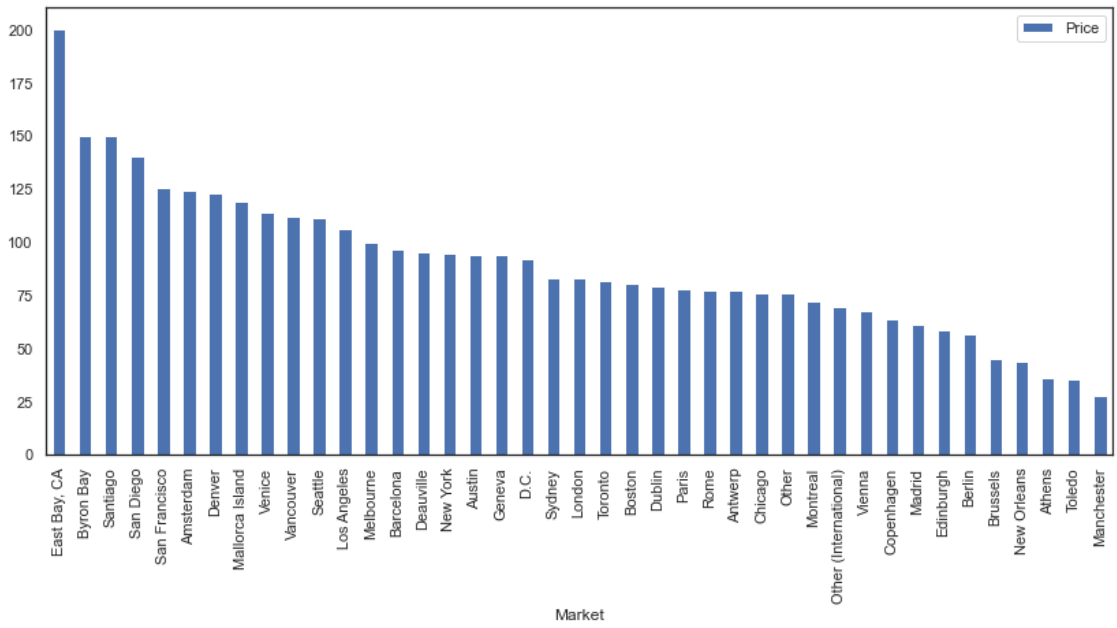
[57]:
```
<AxesSubplot:xlabel='Bed Type'>
```



En este caso no parece haber diferencias excesivas en el precio medio según el tipo de cama.

[58]:
```
plt.rcParams['figure.figsize'] = (14,6)
pd.DataFrame(datos.groupby('Market')['Price'].mean().
  ↪sort_values(ascending=False)).plot(kind='bar')
```

[58]: <AxesSubplot:xlabel='Market'>



Observando el precio medio en función de las ciudades en el dataset observamos unos extremos curiosamente lejanos a la situación del mercado de la vivienda. Esto se puede deber a que las legislaciones locales sobre la plataforma Airbnb o el turismo son factores que no afectan tanto a la hora de comprar una vivienda, pero que alteran drásticamente el rango de precios de los alojamientos en esta plataforma.

7 7. Encoding

Respecto a las variables categóricas, haremos un encoding para finalmente poder trabajar con todas las variables juntas.

Tanto "Property Type", como "Room Type" y "Bed Type" las codificamos de forma binaria con OneHotEncoder.

La variable "Market" la codificamos mediante CatBoostEncoder, ya que contiene muchos valores y cada uno tendrá un peso distinto

[59]:
```
ohe = OneHotEncoder()
oh_array = ohe.fit_transform(datos['Property Type'].values.reshape(-1, 1)).
↪toarray()
oh_df = pd.DataFrame(oh_array,columns=['Apartment','House','Other'])
datos = pd.concat([datos,oh_df],axis=1)
datos = datos.drop(columns='Property Type')
```

[60]:
```
ohe = OneHotEncoder()
oh_array = ohe.fit_transform(datos['Room Type'].values.reshape(-1, 1)).toarray()
oh_df = pd.DataFrame(oh_array,columns=['Entire home/apt','Private room','Shared␣
↪room'])
datos = pd.concat([datos,oh_df],axis=1)
datos = datos.drop(columns='Room Type')
```

[61]:
```
ohe = OneHotEncoder()
oh_array = ohe.fit_transform(datos['Bed Type'].values.reshape(-1, 1)).toarray()
oh_df = pd.DataFrame(oh_array,columns=['Airbed','Couch','Futon','Pull-out␣
↪Sofa','Real Bed'])
datos = pd.concat([datos,oh_df],axis=1)
datos = datos.drop(columns='Bed Type')
```

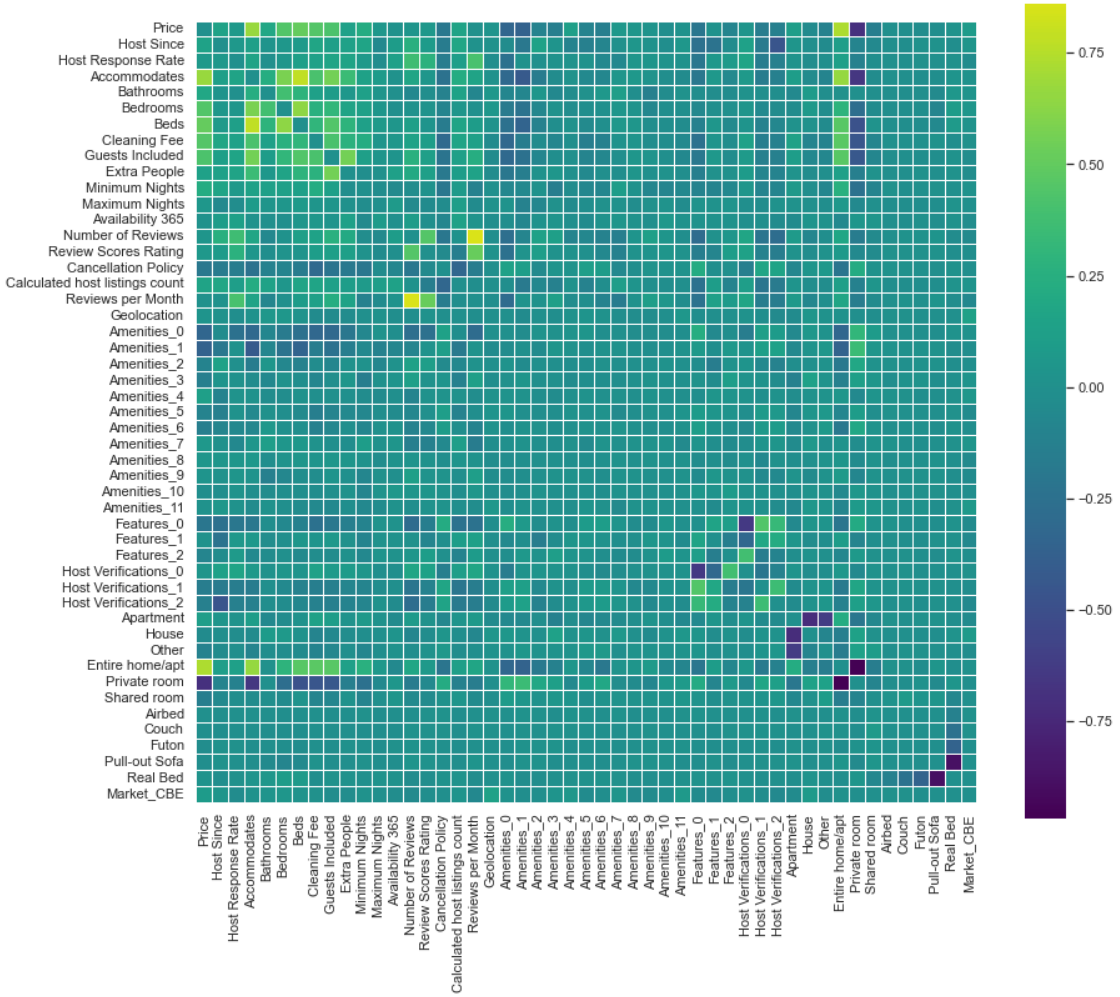
[62]:
```
cbe = ce.CatBoostEncoder(cols=['Market'])
datos['Market_CBE'] = cbe.fit_transform(datos['Market'],datos.Price)
datos = datos.drop(columns='Market')
```

[63]:
```
datos.head()
```

[63]:
```
   Price  Host Since  Host Response Rate  Accommodates  Bathrooms  Bedrooms  \
0   58.0       421.0                100.0             2        1.0       1.0
1   70.0       150.0                100.0             4        1.0       1.0
2   70.0       150.0                100.0             4        1.0       1.0
3  149.0       451.0                 99.0             5        1.0       2.0
4  106.0       451.0                 99.0             6        1.0       2.0

   Beds  Cleaning Fee  Guests Included  Extra People  …  Other  \
0   1.0           0.0                1             0  …    0.0
1   1.0          40.0                2            44  …    0.0
2   1.0          60.0                2            42  …    0.0
3   2.0          55.0                1             0  …    0.0
4   3.0           0.0                1             0  …    0.0

   Entire home/apt  Private room  Shared room  Airbed  Couch  Futon  \
0              1.0           0.0          0.0     0.0    0.0    0.0
1              1.0           0.0          0.0     0.0    0.0    0.0
2              1.0           0.0          0.0     0.0    0.0    0.0
3              1.0           0.0          0.0     0.0    0.0    0.0
4              1.0           0.0          0.0     0.0    0.0    0.0

   Pull-out Sofa  Real Bed  Market_CBE
0            0.0       1.0   63.710635
1            0.0       1.0   63.710635
2            0.0       1.0   66.855318
3            0.0       1.0   67.903545
4            0.0       1.0   88.177659

[5 rows x 49 columns]
```

Buscando reducir más el número de variables, realizamos un gráfico de correlación y eliminamos las variables con alta correlación.

[64]:
```
funciones_auxiliares.get_corr_matrix(dataset = datos, metodo='spearman',␣
↪size_figure=[14,12])
```



[64]: 0

[65]:
```
datos = datos.drop(columns=['Accommodates','Entire home/apt','Pull-out Sofa'])
```

[66]:
```
datos.head()
```

[66]:

	Price	Host Since	Host Response Rate	Bathrooms	Bedrooms	Beds	\
0	58.0	421.0	100.0	1.0	1.0	1.0	
1	70.0	150.0	100.0	1.0	1.0	1.0	
2	70.0	150.0	100.0	1.0	1.0	1.0	
3	149.0	451.0	99.0	1.0	2.0	2.0	
4	106.0	451.0	99.0	1.0	2.0	3.0	

	Cleaning Fee	Guests Included	Extra People	Minimum Nights	…	\
0	0.0	1	0	2	…	
1	40.0	2	44	1	…	

```
2          60.0                2            42                1  …
3          55.0                1             0                2  …
4           0.0                1             0                2  …

   Apartment  House  Other  Private room  Shared room  Airbed  Couch  Futon  \
0        1.0    0.0    0.0           0.0          0.0     0.0    0.0    0.0
1        1.0    0.0    0.0           0.0          0.0     0.0    0.0    0.0
2        1.0    0.0    0.0           0.0          0.0     0.0    0.0    0.0
3        1.0    0.0    0.0           0.0          0.0     0.0    0.0    0.0
4        1.0    0.0    0.0           0.0          0.0     0.0    0.0    0.0

   Real Bed  Market_CBE
0       1.0   63.710635
1       1.0   63.710635
2       1.0   66.855318
3       1.0   67.903545
4       1.0   88.177659

[5 rows x 46 columns]
```

8 8. Conjuntos Train y Test

Finalizada toda la limpieza y preprocesamiento de los datos procedemos a crear los conjuntos de entrenamiento y test que usaremos en los modelos predictivos que vamos a estudiar. También escalaremos los conjuntos en previsión de los modelos que lo requieran.

[67]:
```
X = datos.iloc[:,1:46]
Y = datos['Price'].astype('int')

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=0)

X_train.shape, X_test.shape
```

[67]:
```
((11440, 45), (2860, 45))
```

[68]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
model_scaled = scaler.fit(X_train)
X_train_scaled = pd.DataFrame(scaler.transform(X_train), columns=X_train.
 ↪columns, index=X_train.index)
X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns,␣
 ↪index=X_test.index)
```

9 9. Modelos de Regresión

9.0.1 9.1 KNN

Vamos a utilizar el algoritmo KNN para regresión.

Este método consiste en, una vez identificados los k vecinos de cada punto, en lugar de considerar su clase y establecer un sistema de "votación", se considerará el valor que toma la etiqueta para cada uno de ellos y se devolverá como predicción el valor medio de dichos valores.

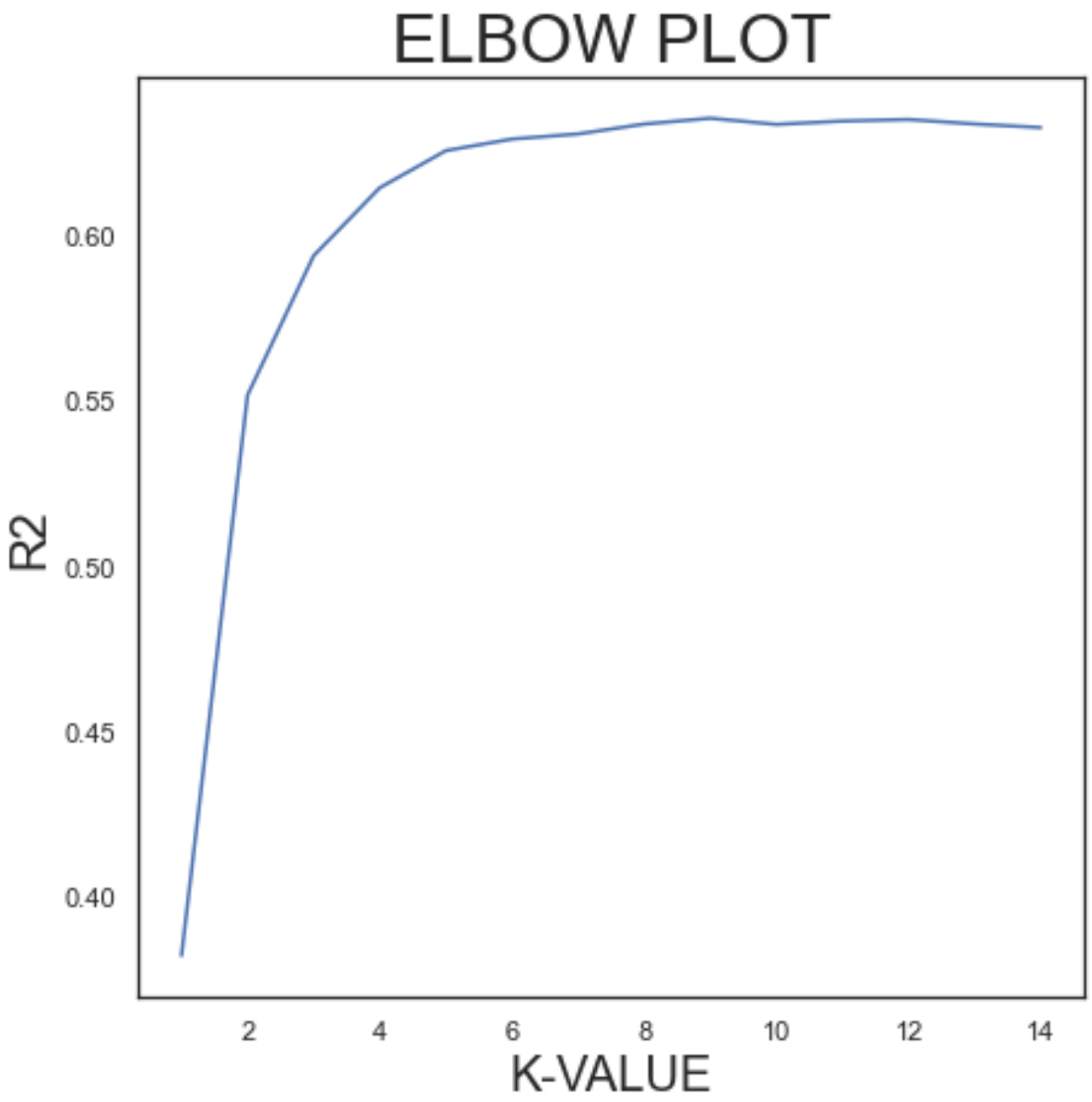
En primer lugar vamos a buscar el k óptimo para el que aplicar el algoritmo, con un gráfico de codo, en el que veremos que k ofrece el menor error cuadrático medio.

[69]:
```
k_list=np.arange(1,15,1)
knn_dict={} # Guardamos los k
for i in k_list:

# Creamos el modelo
    knn=KNeighborsRegressor(n_neighbors=int(i))
    model_knn=knn.fit(X_train_scaled,Y_train)
    y_knn_pred=model_knn.predict(X_test_scaled)

# Guardamos el MSE
    mse=r2_score(Y_test,y_knn_pred)
    knn_dict[i]=mse
```

[70]:
```
# Gráfico de codo:
fig,ax=plt.subplots(figsize=(7,7))
print(knn_dict)
ax.plot(knn_dict.keys(),knn_dict.values())
ax.set_xlabel('K-VALUE', fontsize=20)
ax.set_ylabel('R2' ,fontsize=20)
ax.set_title('ELBOW PLOT' ,fontsize=28)
```

```
{1: 0.38221623405158056, 2: 0.5517244562562571, 3: 0.5938639468200975, 4:
0.6143987636186339, 5: 0.625627006088332, 6: 0.629126500573963, 7:
0.630669222375631, 8: 0.6336620742886443, 9: 0.6354190600037447, 10:
0.6335560431871419, 11: 0.6345717809642561, 12: 0.6350395100518569, 13:
0.6336863804441057, 14: 0.6326119212899843}
```

[70]:
```
Text(0.5, 1.0, 'ELBOW PLOT')
```



Usando Cross-Validation:

[71]:
```
random_grid = {'n_neighbors': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]}
knn_tune = KNeighborsRegressor()
knn_random = GridSearchCV(estimator = knn_tune, \
                          param_grid = random_grid, cv = 3)

knn_random.fit(X_train_scaled,Y_train)
print(knn_random.best_estimator_)
```

```
KNeighborsRegressor(n_neighbors=14)
```

Acabamos de comprobar que tanto por Cross-Validation como por el gráfico de codo que el mejor k es 12, por tanto **usamos k = 12** para KNN Regressor.

[72]:
```
knn = KNeighborsRegressor(n_neighbors=12)

knn.fit(X_train_scaled, Y_train)

knn_pred=knn.predict(X_test_scaled)
```

Vemos las puntuaciones asociadas al modelo con diferentes métricas.

[73]:
```
mse_knn = mean_squared_error(Y_test, knn_pred)
print("Regresión con KNN Regressor (MSE):",mse_knn )
rmse_knn = mean_squared_error(Y_test, knn_pred,squared=False)
print("Regresión con KNN Regressor (RMSE):",rmse_knn)
r2_knn = r2_score(Y_test, knn_pred)
print("Regresión con KNN Regressor (R^2):", r2_knn)
mae_knn =  mean_absolute_error(Y_test, knn_pred)
print("Regresión con KNN Regressor (MAE):",mae_knn)
```

```
Regresión con KNN Regressor (MSE): 597.2908993783993
Regresión con KNN Regressor (RMSE): 24.439535580251917
Regresión con KNN Regressor (R^2): 0.6350395100518569
Regresión con KNN Regressor (MAE): 16.63648018648019
```

9.0.2 9.2 Regresión Lineal

El modelo de regresión lineal busca determinar si existe una relación lineal entre una variable dependiente y una o varias independientes. El modelo debe examinar cuales son nuestras variables más significativas para predecir el precio del alojamiento. Por ejemplo parece lógico pensar que cuanto más espacioso sea un alojamiento mayor será su precio.

Además se pueden aplicar regularizaciones al modelo, que consiste en añadir penalizaciones a la función de costo, de manera que la complejidad del modelo disminuye a la vez que minimizamos la función de coste. Esto resulta en modelos más simples que tienden a generalizar mejor.

La regularización que usaremos es Lasso(L1) ya que favorece que algunos de los coeficientes acaben valiendo 0. Esto será de ayuda para descubrir cuáles de nuestras variables independientes son relevates para el precio del alojamieto.

Empezaremos buscando el mejor hiperparámetro 'alpha' usando cross validation

[74]:
```
#Espacio de búsqueda del hiperparámetro alpha
# ==============================================================================
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=123)
lasso_alphas = 10**np.linspace(-2,2,200)
# ==============================================================================
#Búsqueda LassoCV
# ==============================================================================
lasso = LassoCV(alphas=lasso_alphas, cv=cv, n_jobs=-1)
lasso.fit(X_train_scaled, Y_train)
#Resultados
# ==============================================================================
```

```
print('alpha:', lasso.alpha_)
```

```
alpha: 0.10595601792776159
```

Por lo tanto construiremos nuestro modelo tomando este alpha.

[75]:
```
modelLasso=lasso.fit(X_train_scaled, Y_train)
trainPred_lasso = modelLasso.predict(X_train_scaled)
testPred_lasso =modelLasso.predict(X_test_scaled)
```

Observamos los coeficientes que hemos obtenido.

[76]:
```
lasso_betas = list(zip(X_train_scaled.columns,modelLasso.coef_))
coeff= pd.DataFrame(lasso_betas,columns=['característica','coeficiente'])
coeff.sort_values(by='coeficiente', inplace=True)
coeff
```

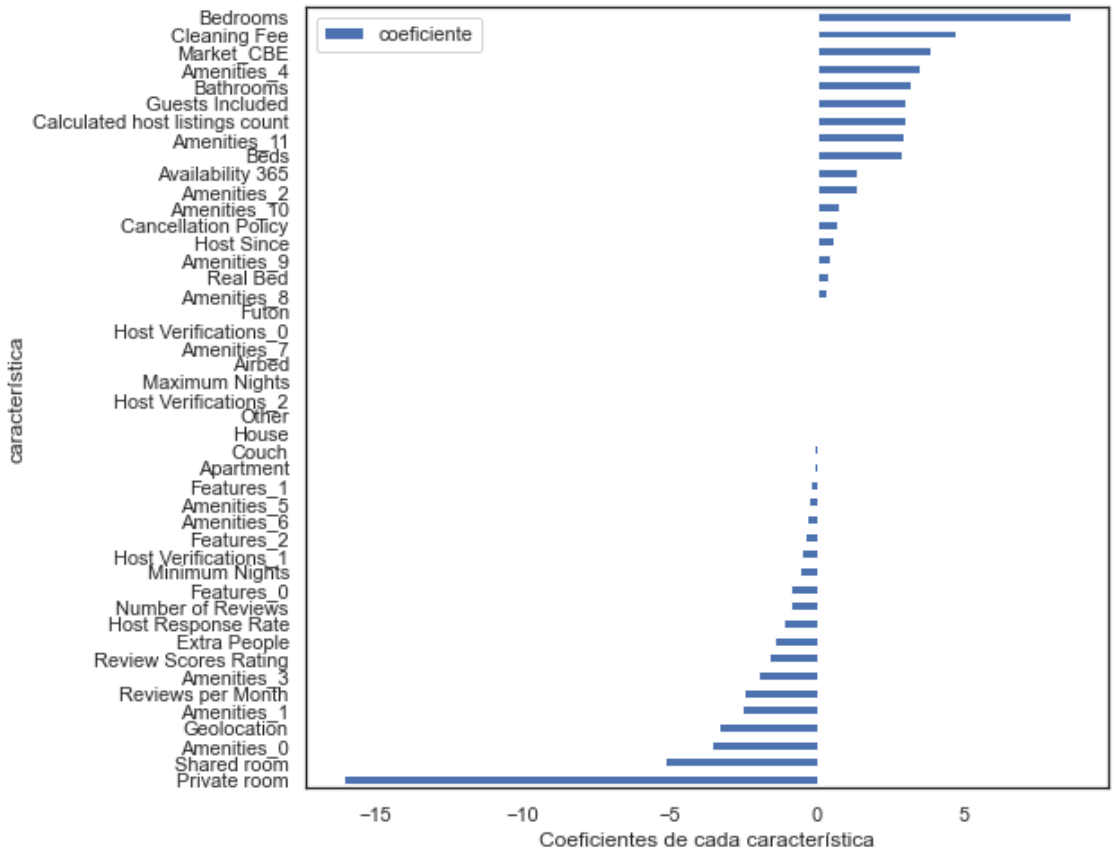
[76]:

	característica	coeficiente
38	Private room	-16.096930
39	Shared room	-5.182632
17	Amenities_0	-3.549029
16	Geolocation	-3.326399
18	Amenities_1	-2.519007
15	Reviews per Month	-2.493212
20	Amenities_3	-2.015342
12	Review Scores Rating	-1.624376
7	Extra People	-1.456256
1	Host Response Rate	-1.102349
11	Number of Reviews	-0.888577
29	Features_0	-0.857894
8	Minimum Nights	-0.557222
33	Host Verifications_1	-0.517025
31	Features_2	-0.410492
23	Amenities_6	-0.351619
22	Amenities_5	-0.294745
30	Features_1	-0.199687
35	Apartment	-0.092461
41	Couch	-0.090153
36	House	0.000000
37	Other	0.000000
34	Host Verifications_2	0.000000
9	Maximum Nights	-0.000000
40	Airbed	0.000000
24	Amenities_7	-0.000000
32	Host Verifications_0	0.000000
42	Futon	-0.000000
25	Amenities_8	0.332257
43	Real Bed	0.397563

```
26                       Amenities_9     0.471462
0                         Host Since     0.585640
13               Cancellation Policy     0.682682
27                      Amenities_10     0.736194
19                       Amenities_2     1.342681
10                  Availability 365     1.383217
4                               Beds     2.895764
28                      Amenities_11     2.951738
14  Calculated host listings count     3.029065
6                    Guests Included     3.037075
2                          Bathrooms     3.219869
21                       Amenities_4     3.487255
44                        Market_CBE     3.869567
5                       Cleaning Fee     4.708103
3                           Bedrooms     8.652683
```

Las variables que no son significativas son “House”, “Other”, “Maximun Nights”, “Airbed”, “Host Verifications_0”, “Apartment” y “Futon”. Por otra parte las 5 más importantes son “Private Room”, “Bedrooms”, “Shared room” y “Cleaning Fee”.

[77]:
```
coeff2 = coeff.set_index('característica', drop=True)
coeff2.plot.barh(figsize=(8,8))
plt.xlabel('Coeficientes de cada característica')
plt.show()
```



[78]:
```
mse_lasso = mean_squared_error(Y_test, testPred_lasso)
print("Regresión con Lasso (MSE):",mse_lasso)
rmse_lasso=mean_squared_error(Y_test, testPred_lasso,squared=False)
print("Regresión con Lasso (RMSE):", rmse_lasso)
r2_lasso = r2_score(Y_test, testPred_lasso)
print("Regresión con Lasso (R^2):", r2_lasso)
mae_lasso = mean_absolute_error(Y_test, testPred_lasso)
print("Regresión con Lasso (MAE):", mae_lasso)
```

```
Regresión con Lasso (MSE): 650.0554181467473
Regresión con Lasso (RMSE): 25.49618438407495
Regresión con Lasso (R^2): 0.6027989976957921
Regresión con Lasso (MAE): 17.936756471835313
```

9.0.3 9.3 Random forest

Un árbol de decisión es una técnica de aprendizaje automático que se puede utilizar para regresión y clasificación. Los árboles de decisión tienen la tendencia de sobre-ajustar (overfit). Esto quiere decir que tienden a aprender muy bien los datos de entrenamiento pero su generalización no es tan buena.

Podemos mejorar la capaidad de los árboles combinando varios. A esta combinación de la llama Random Forest.

Al combinar sus resultados, unos errores se compensan con otros y tenemos una predicción que generaliza mejor.

Este algoritmo puede capturar relaciones no lineales y no requiere ninguna transformación de las características si se trata de datos no lineales ya que los árboles de decisión no tienen en cuenta múltiples combinaciones ponderadas simultáneamente. Por tanto esta vez no escalaremos el set de entrenamiento. Es un algoritmo rápido y eficiente en comparación con KNN y ofrece fácil interpretación y visualización.

Vamos a ver cuáles son los parámetros óptimos para Random Forest con cross validation.

[79]:
```
random_grid = {'n_estimators': [200],
               'max_features': [2,5],
              'max_depth': [40,70],
              'min_samples_split': [40,50],
               'max_leaf_nodes':[50,70],
               'max_features': [2,5]}
rf_tune = RandomForestRegressor()
rf_random = GridSearchCV(estimator = rf_tune, param_grid = random_grid, cv = 3,␣
 ↪verbose=2,n_jobs = 2)

rf_random.fit(X_train,Y_train)
print(rf_random.best_estimator_)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
RandomForestRegressor(max_depth=40, max_features=5, max_leaf_nodes=70,
                      min_samples_split=40, n_estimators=200)
```

[80]:
```
# Creamos el modelo:

RF = RandomForestRegressor(max_depth=40, max_features=5, max_leaf_nodes=70,
                       min_samples_split=50, n_estimators=200)
RF.fit(X_train,Y_train)

trainPred = RF.predict(X_train)
forest_pred =RF.predict(X_test)
```

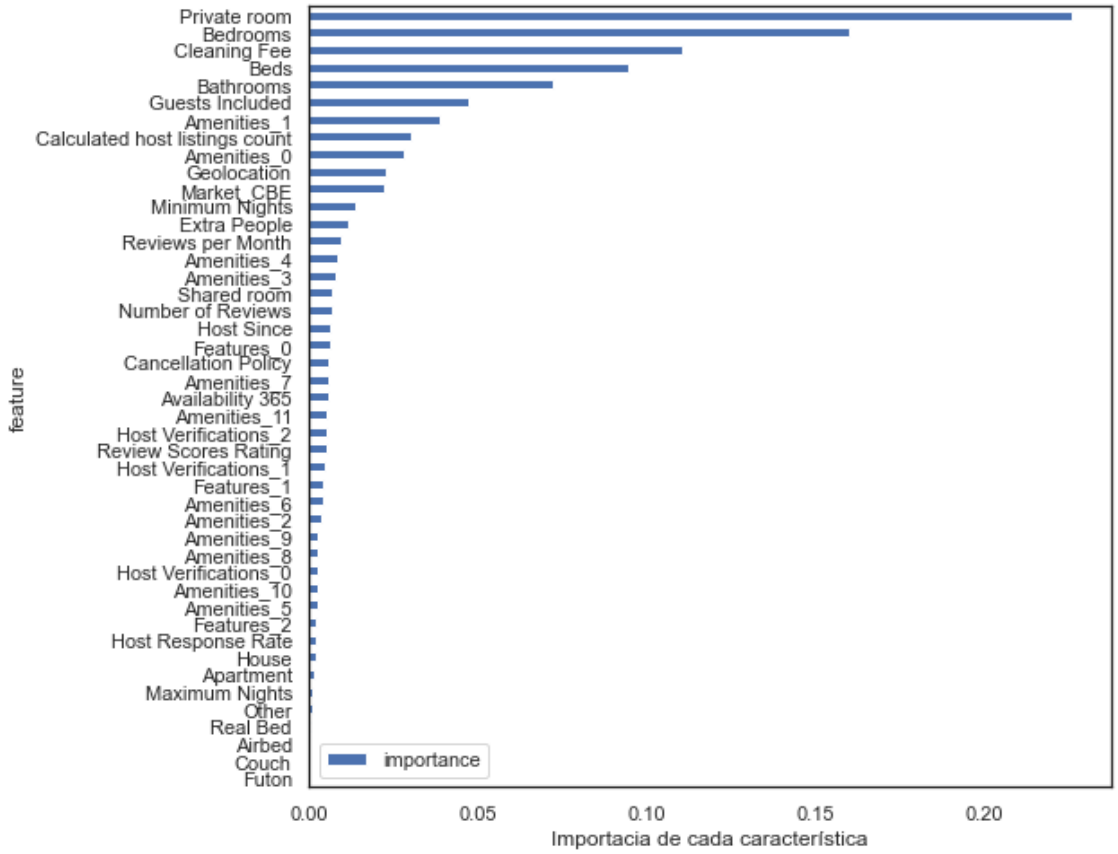
Vemos cuáles son los pesos o importancias de cada variable en la regresión.

[81]:
```
importancesRF = RF.feature_importances_
feat_imp1 = pd.DataFrame(importancesRF, columns=['Weight'], index=X_train.
 ↪columns)
feat_imp1.sort_values('Weight', inplace=True)
feat_imp1
```

[81]:

	Weight
Futon	0.000000
Couch	0.000000
Airbed	0.000000
Real Bed	0.000105
Other	0.000642
Maximum Nights	0.001150
Apartment	0.001586
House	0.001791
Host Response Rate	0.001939
Features_2	0.002085
Amenities_5	0.002327
Amenities_10	0.002381
Host Verifications_0	0.002406
Amenities_8	0.002516
Amenities_9	0.002601
Amenities_2	0.003680
Amenities_6	0.004198
Features_1	0.004328
Host Verifications_1	0.004665
Review Scores Rating	0.005139
Host Verifications_2	0.005151
Amenities_11	0.005363
Availability 365	0.005457
Amenities_7	0.005472
Cancellation Policy	0.005899
Features_0	0.006358
Host Since	0.006378
Number of Reviews	0.006537
Shared room	0.006769
Amenities_3	0.007828
Amenities_4	0.008557
Reviews per Month	0.009138
Extra People	0.011582
Minimum Nights	0.013783
Market_CBE	0.022309
Geolocation	0.022612
Amenities_0	0.028055
Calculated host listings count	0.029907
Amenities_1	0.038796
Guests Included	0.047411
Bathrooms	0.072163
Beds	0.094553
Cleaning Fee	0.110251
Bedrooms	0.159946
Private room	0.226183

[82]:
```
feat_imp = pd.DataFrame({'importance':RF.feature_importances_})
feat_imp['feature'] = X_train.columns
feat_imp.sort_values(by='importance', ascending=False, inplace=True)

feat_imp.sort_values(by='importance', inplace=True)
feat_imp = feat_imp.set_index('feature', drop=True)
feat_imp.plot.barh(figsize=(8,8))
plt.xlabel('Importacia de cada característica')
plt.show()
```



Vemos las puntuaciones asociadas al modelo con diferentes métricas.

[83]:
```
mse_rf = mean_squared_error(Y_test, forest_pred)
print("Regresión con Random Forest (MSE):", mse_rf)
rmse_rf = mean_squared_error(Y_test, forest_pred,squared=False)
print("Regresión con Random Forest (RMSE):", rmse_rf)
r2_rf = r2_score(Y_test, forest_pred)
print("Regresión con Random Forest (R^2):", r2_rf)
mae_rf = mean_absolute_error(Y_test, forest_pred)
print("Regresión con Random Forest (MAE):", mae_rf)
```

```
Regresión con Random Forest (MSE): 593.1401086305395
Regresión con Random Forest (RMSE): 24.35446793979576
Regresión con Random Forest (R^2): 0.6375757526542264
Regresión con Random Forest (MAE): 17.286058715050096
```

9.0.4 9.4 Algoritmo SVM en Regresión

SVR (Support Vector Regression) es una variación de SVM (utilizado para clasificar), sin embargo, con esta variante el algoritmo SVR se utiliza para predecir valores con un margen de tolerancia (épsilon) establecido por el investigador.

Veamos cuál es el épsilon óptimo para SVR con cross validation.

[84]:
```
# Grid de hiperparámetros
# ==============================================================================
param = {'epsilon':[1,5,10,15,20,25]}
modelsvr = SVR(kernel='linear')

# Búsqueda por cross validation
# ==============================================================================
grid = GridSearchCV(modelsvr,
                    param,
                    cv=5)
_=grid.fit(X_train_scaled,Y_train)
grid.best_estimator_
print(grid.best_estimator_)
```

```
SVR(epsilon=20, kernel='linear')
```

Por lo tanto construiremos nuestro modelo tomando epsilon con valor 20.

[85]:
```
modelSVR = SVR(kernel='linear',epsilon=20)
modelSVR.fit(X_train_scaled,Y_train)

trainPred_svr = modelSVR.predict(X_train_scaled)
testPred_svr =modelSVR.predict(X_test_scaled)
```

Veamos cuales son los pesos de cada variable en la regresión.

[86]:
```
mse_svr = mean_squared_error(Y_test, testPred_svr)
print("Regresión con SVR (MSE):",mse_svr)
rmse_svr=mean_squared_error(Y_test, testPred_svr,squared=False)
print("Regresión con SVR (RMSE):", rmse_svr)
r2_svr = r2_score(Y_test, testPred_svr)
print("Regresión con SVR (R^2):", r2_svr)
mae_svr = mean_absolute_error(Y_test, testPred_svr)
print("Regresión con SVR (MAE):", mae_svr)
```

```
Regresión con SVR (MSE): 648.5770078263057
Regresión con SVR (RMSE): 25.467175104952368
```

```
Regresión con SVR (R^2): 0.6037023453869328
Regresión con SVR (MAE): 17.810014113652578
```

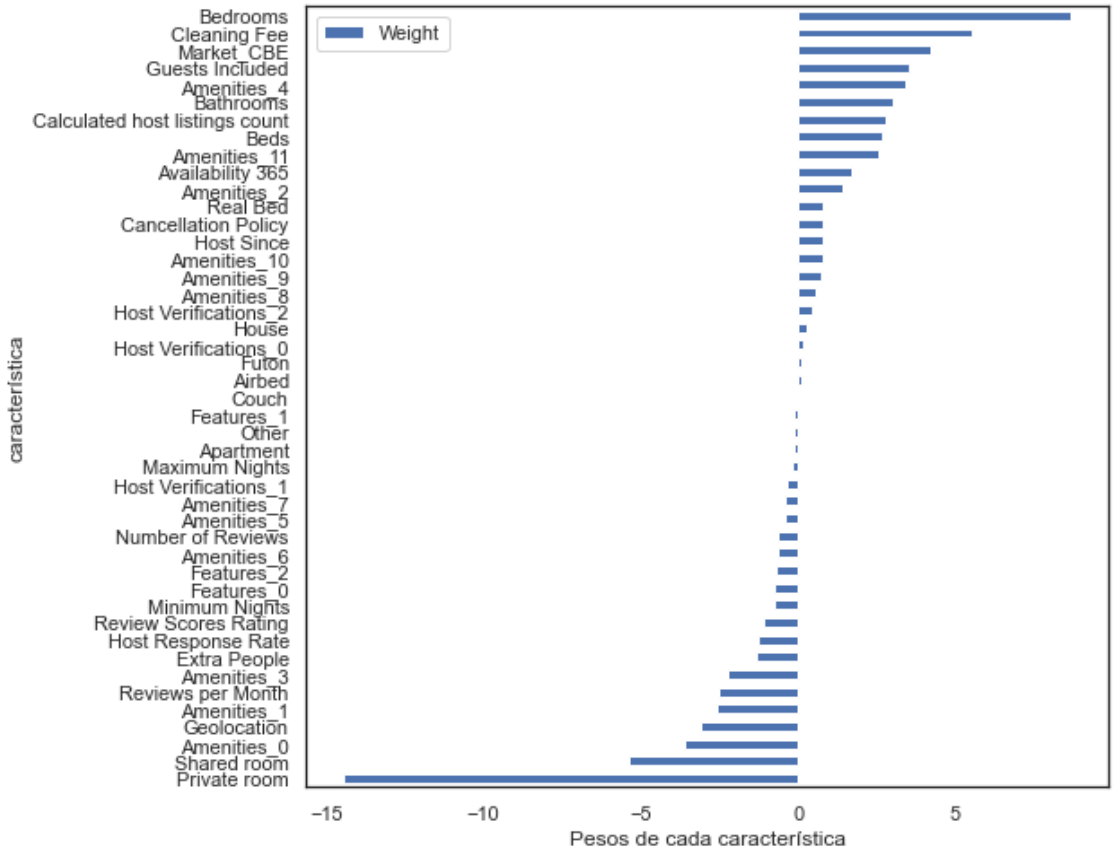
[87]:
```
coefSVR = modelSVR.coef_
feat_imp = pd.DataFrame({'Weight':coefSVR[0]})
feat_imp['característica'] = X_train.columns
feat_imp.sort_values(by='Weight', inplace=True)
feat_imp
```

[87]:

	Weight	característica
38	-1.447276e+01	Private room
39	-5.370641e+00	Shared room
17	-3.616856e+00	Amenities_0
16	-3.120183e+00	Geolocation
18	-2.594996e+00	Amenities_1
15	-2.535945e+00	Reviews per Month
20	-2.218168e+00	Amenities_3
7	-1.341212e+00	Extra People
1	-1.257330e+00	Host Response Rate
12	-1.081150e+00	Review Scores Rating
8	-7.348406e-01	Minimum Nights
29	-7.284367e-01	Features_0
31	-7.094155e-01	Features_2
23	-6.536099e-01	Amenities_6
11	-6.130485e-01	Number of Reviews
22	-4.392648e-01	Amenities_5
24	-3.984282e-01	Amenities_7
33	-3.511861e-01	Host Verifications_1
9	-1.811975e-01	Maximum Nights
35	-1.514850e-01	Apartment
37	-1.123083e-01	Other
30	-9.804164e-02	Features_1
41	4.618528e-14	Couch
40	9.305979e-02	Airbed
42	9.874830e-02	Futon
32	1.600009e-01	Host Verifications_0
36	2.938642e-01	House
34	4.473901e-01	Host Verifications_2
25	5.383670e-01	Amenities_8
26	7.335611e-01	Amenities_9
27	7.739927e-01	Amenities_10
0	7.764569e-01	Host Since
13	7.844151e-01	Cancellation Policy
43	7.846787e-01	Real Bed
19	1.400682e+00	Amenities_2
10	1.713061e+00	Availability 365
28	2.577613e+00	Amenities_11

```
4    2.685842e+00                             Beds
14   2.787908e+00  Calculated host listings count
2    3.016446e+00                        Bathrooms
21   3.389496e+00                      Amenities_4
6    3.519683e+00                  Guests Included
44   4.207692e+00                       Market_CBE
5    5.539054e+00                     Cleaning Fee
3    8.687086e+00                         Bedrooms
```

[88]:
```
feat_imp = feat_imp.set_index('característica', drop=True)
feat_imp.plot.barh(figsize=(8,8))
plt.xlabel('Pesos de cada característica')
plt.show()
```



10 10. Resultados de los Modelos de Regresión y Conclusión

[89]:
```
resultados = pd.DataFrame({
    'Modelo':['KNN','Lasso','Random Forest','SVR'],
    'MSE':[mse_knn,mse_lasso,mse_rf,mse_svr],
    'RMSE':[rmse_knn,rmse_lasso,rmse_rf,rmse_svr],
    'R2':[r2_knn,r2_lasso,r2_rf,r2_svr],
    'MAE':[mae_knn,mae_lasso,mae_rf,mae_svr]
    })
resultados
```

[89]:

	Modelo	MSE	RMSE	R2	MAE
0	KNN	597.290899	24.439536	0.635040	16.636480
1	Lasso	650.055418	25.496184	0.602799	17.936756
2	Random Forest	593.140109	24.354468	0.637576	17.286059
3	SVR	648.577008	25.467175	0.603702	17.810014

A la hora de medir el grado de acierto de los diferentes modelos de regresión que hemos obtenido deberemos tomar en cuanta los valores obtenidos en sus métricas, para hacernos una idea de los acertado que sería el modelo que hemos construido.

El error cuadrático medio (Mean Squared Error, MSE), también llamado mínimos cuadrados ordinarios (Ordinary Least Squares, OLS) calcula la media de los errores al cuadrado y cuanto mayor sea, peor es el modelo. Se debe tener en cuenta que al medir un valor cuadrático penaliza los errores mayores y nunca es negativo, esto ayudará a identificar resultados atípicos. Esta métrica no funciona bien con datos de baja calidad, pero afortunadamente toda la fase de preprocessing ha dado sus frutos y ese no es un problema que nos afecte.

La raíz cuadrada del error cuadrático medio (Root Mean Squared Error, RMSE), es equivalente a la métrica anterior en términos de ordenar cómo funcionan los modelos pero al ser una raíz cuadrada tiene el mismo orden de magnitud que la variable objetivo.

El error absoluto medio (Mean Absolute Error, MAE) se calcula como el promedio de los errores en valor absoluto. Al no ser cuadrático, penaliza menos los errores altos y atípicos que el MSE.

El coeficiente de determinación o R^2, compara el error cuadrático medio de mi modelo con un error cuadrático medio de referencia. Esta referencia se calcula asumiendo que el valor de una predicción es el promedio de todas las estimaciones. Es decir, sin tener en cuenta ningún modelo ni dependencia con otras variables En general el R^2 se puede ver como una versión estandarizada del error cuadrático medio que refleja la relación de la variable objetivo con respecto al resto de variables del modelo. Cuanto más se acerque a 1, mayor es la relación de las variables del modelo con el objetivo. Lo que quiere decir que mejor es el resultado del algoritmo.

Observando las métricas no detectamos valores de MSE, RMSE y MAE que nos avisen de que nuestros modelos son demasiado erróneos. Tomando como métrica principal R^2 observamos que siendo el máximo valor posible 1 los valores de los modelos son bastante decentes, estando todos por encima del 0'6.

Finalmente, tras visualizar, limpiar, preprocesar y aplicar los algoritmos con cuidado, hemos logrado producir varios modelos de regresión de aprendizaje automático con puntuaciones R^2 de predicción

relativamente altas. El mejor modelo hasta el momento fue un modelo KNN que obtuvo un valor R^2 de 0.636 seguido de cerca por el modelo Ramdom Forest que obtuvo un valor R^2 de 0.634. Con cualquiera de estos dos modelos ya sería viable construir un evaluador automático de precios a partir de las carácterísticas de los alojamientos de Airbnb y por tanto se puede considerar el objetivo de la práctica cumplido.