

Análisis, visualización, exploración y modelado de los datos de Airbnb en Madrid: determinación de la relación entre precio y variables predictivas



Memoria proyecto final

Introducción a la programación & Big Data

KeepCoding Tech. School

15 febrero 2023

Autoras:

Apellidos, Nombre
Baluja García, Tania
Jiménez-Alfaro Ayala, Ana
Jiménez González, Laura
Manrique Savasta, Emely
Montero Martínez, Liliana

ÍNDICE

	Pp.
OBJETIVO	1
RESUMEN	1
CONSIDERACIONES PREVIAS	1
COMPRESIÓN Y ESTUDIO INICIAL DE LOS DATOS	3
ARQUITECTURA Y VALIDACIÓN DE LOS DATOS	6
ANÁLISIS EXPLORATORIO DE DATOS	7
PROCESO EN POSTGRESQL	16
VISUALIZACIÓN DE LAS MÉTRICAS	18
PREPROCESAMIENTO Y MODELADO	23
MODELADO PARA ESTIMACIÓN DEL PRECIO	33
CONCLUSIONES	38
BIBLIOGRAFÍA	39

OBJETIVO

El objetivo principal del proyecto es analizar las diferentes variables que afectan al precio de una vivienda en alquiler con el objetivo de poder incluirlas en un algoritmo de regresión lineal capaz de predecir dicho precio. Así como proporcionar un cuadro de mandos o dashboard que facilite la detección de acciones para maximizar la rentabilidad de los alojamientos de los clientes de Airbnb.

RESUMEN

Se ha trabajado con un conjunto de datos que contiene información sobre los alojamientos de Airbnb en la ciudad de Madrid. En el proyecto, se ha llevado a cabo la definición del conjunto de datos, la arquitectura y validación de datos, análisis exploratorio, visualización de métricas y modelado. Se han utilizado técnicas de análisis y modelado para comprender las tendencias y relaciones entre las diferentes variables, además de la aplicación de regresiones lineales para estimar el precio en función de las características de los inmuebles. Con esta información, se espera brindar soluciones eficaces para mejorar la rentabilidad de los alojamientos de los clientes actuales y asesorar a los posibles clientes potenciales.

CONSIDERACIONES PREVIAS

En este apartado, se especifican las suposiciones iniciales con el objetivo de proporcionar un marco de referencia claro y coherente para el análisis posterior de los datos.

- La ausencia de valores en las variables relacionadas con las estancias de los alojamientos se considerará como 0. Es decir: *'Beds'*, *'Bedrooms'* y *'Bathrooms'*
- La ausencia de valor en *'Security_deposit'*, *'Cleaning_fee'*, *'Amenities'* y *'Features'* se considerarán como 0, ya que se asumirá que no aplica a dicho alojamiento.

- Dado que no hay datos del número de veces que se ha alquilado el alojamiento, asumiremos que en el 100% de ocasiones que se alquila una propiedad, se obtiene una 'review', es decir: *1 Review = 1 vez alquilado*.

COMPRENSIÓN Y ESTUDIO INICIAL DE LOS DATOS

El origen de los datos utilizados en el proyecto, provienen de *Opendatasoft*¹ y consta de un archivo en formato .CSV que se corresponde a propiedades turísticas ubicadas en Madrid, España en el portal de Airbnb.

Dicho archivo contiene información recopilada hasta el 2017 y se estructura con 48 columnas y 14.780 filas.

Field	Type	Calculated	Description	Reference
id	integer		Airbnb's unique identifier for the listing	
listing_url	text	y		
scrape_id	bigint	y	Inside Airbnb "Scrape" this was part of	
last_scraped	datetime	y	UTC. The date and time this listing was "scraped".	
source	text		One of "neighbourhood search" or "previous scrape". "neighbourhood search" means that the listing was found by searching the city, while "previous scrape" means that the listing was seen in another scrape performed in the last 65 days, and the listing was confirmed to be still available on the Airbnb site.	
name	text		Name of the listing	
description	text		Detailed description of the listing	
neighborhood_overview	text		Host's description of the neighbourhood	
picture_url	text		URL to the Airbnb hosted regular sized image for the listing	
host_id	integer		Airbnb's unique identifier for the host/user	
host_url	text	y	The Airbnb page for the host	
host_name	text		Name of the host. Usually just the first name(s).	

Figura 1. Fragmento de Data Dictionary for listings.csv detailed file² (Inside Airbnb)

Una vez lograda la familiarización con el data set, se realiza la importación del fichero mediante lenguaje Python. Usando el mismo lenguaje, se comienza una primera limpieza de los datos y tareas básicas como eliminación de duplicados, corrección de valores erróneos, tratamiento de valores nulos y conversión de tipo de datos para asegurarse de que la información se encuentre en un formato adecuado para su posterior análisis.

1. Selección de columnas

En un primer lugar, se hace una selección inicial de columnas aspirantes para el análisis. Se suprimen variables que no son consideradas como relevantes en este análisis y aquellas con alto contenido en valores nulos (de ahora en adelante NaNs).

En este caso, se seleccionan las columnas {ID, Host ID, Host Since, Host Location, Host Response Time, Host Response Rate, Host Listings Count, Host Total Listings Count, Neighbourhood Cleansed, Neighbourhood Group Cleansed, City, State, Zipcode, Market, Smart Location, Country Code, Country, Latitude, Longitude, Property Type, Room Type, Accommodates, Bathrooms, Bedrooms, Beds, Bed Type, Amenities, Square Feet, Price, Weekly Price, Monthly Price, Security Deposit, Cleaning Fee, Guests Included, Extra People (\$), Minimum Nights, Maximum Nights, Number of Reviews, Review Scores Rating, Review Scores Accuracy, Review Scores Cleanliness, Review Scores Checkin, Review Scores Communication, Review Scores Location, Review Scores Value, Cancellation Policy, Calculated host listings count, Reviews per Month, Features}.

2. Limpieza de valores nulos

Como se menciona en el apartado de 'Consideraciones previas', en el caso de las columnas relacionadas con las partes de la vivienda, los valores NaN de las mismas serán sustituidos por ceros. En este caso, se hayan:

- *Bedrooms*: 25
- *Beds*: 49
- *Bathrooms*: 55

Lo mismo sucede con las variables '*Security_deposit*': 8524, '*Cleaning_fee*': 6093, '*Amenities*':170 y '*Features*':1.

3. Comprobación y transformación de formatos a ASCII

Con el objetivo de evitar errores en el tratamiento y análisis de los datos, se comprueban los formatos de todas las variables de data set y se transforman en fecha aquellas variables con datos de fecha que no hayan sido correctamente importados en el archivo, por ejemplo, el caso de “*Host_Since*”, “*First_review*” y “*Last_review*”.

```
df_airbnb.dtypes # las que son de tipo character o string en pandas me las marca como object que indica que pueden ser str o mixed. --> OK
df_airbnb['Host Since'] = pd.to_datetime(df_airbnb['Host Since'], format='%Y-%m-%d')
df_airbnb['First Review'] = pd.to_datetime(df_airbnb['First Review'], format='%Y-%m-%d')
df_airbnb['Last Review'] = pd.to_datetime(df_airbnb['Last Review'], format='%Y-%m-%d')

df_airbnb['Host Since'].dtypes #Compruebo --> dtype('<M8[ns]')
```

Posteriormente, se procede a limpieza de los nombres de columnas, sustituyendo los espacios por “_”, para evitar errores de código, y transformarlos en mayúsculas para una mejor lectura y tratamiento en posteriores herramientas.

Además, se transforma y normalizan los datos de tipo “object” a formato ASCII (unidecode) para mayor compatibilidad con otros lenguajes de programación y aplicaciones software, y con la intención de conseguir mayor agregación de los datos. También, se transforman todos los registros a mayúsculas, para una mayor homogeneidad de los datos, y nuevamente, prevención de errores tipo “case sensitive”.

```
#Transformo las columnas nuevas de First review y last review a datetime64
df_airbnb['LAST_REVIEW'] = pd.to_datetime(df_airbnb['LAST_REVIEW'])
df_airbnb['FIRST_REVIEW'] = pd.to_datetime(df_airbnb['FIRST_REVIEW'])

df_airbnb_string_cols = df_airbnb.select_dtypes(include=['object'], exclude=['datetime64[ns]', 'int64', 'float64'])
#Me seguía cogiendo Host Since y otras variables numéricas incluso al haberle cambiado de tipo, lo excluyo a proposito para asegurarme

df_airbnb_string_cols = df_airbnb_string_cols.astype(str)
df_airbnb_string_cols = df_airbnb_string_cols.applymap(lambda x: unidecode(x))
df_airbnb_string_cols = df_airbnb_string_cols.applymap(lambda x: x.upper())
df_airbnb[df_airbnb_string_cols.columns] = df_airbnb_string_cols

print('***RECuento DE COLUMNAS AFECTADAS CON LA LIMPIA***) #me aseguro que las variables numericas/fechas no están aquí
print(df_airbnb_string_cols.columns)
```

ARQUITECTURA Y VALIDACIÓN DE DATOS

En este apartado trataremos de garantizar la integridad y la calidad de los datos utilizados en el proyecto. Se describirá el proceso de adquisición y almacenamiento de los datos, así como la validación realizada para asegurarse de que los datos son precisos y fiables.

En respuesta a la necesidad del negocio y resultado del previo análisis exploratorio en Python se elaboró un modelo de entidad-relación ER que permite la comprensión del diseño de la base de datos. El diagrama realizado para el proyecto se presenta en la figura II.

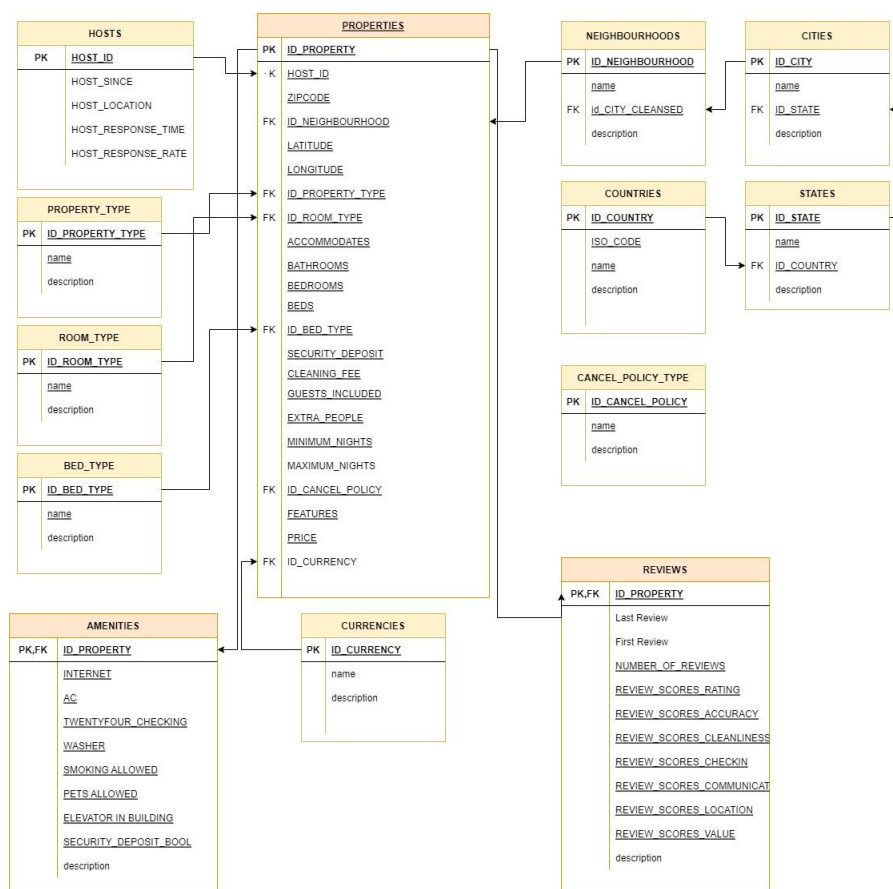


Figura 2: Diagrama entidad relación.

Obsérvese que la entidad principal corresponde a la gestión de las propiedades y que está se relaciona con otras entidades. Un Host puede tener n propiedades y se considera oportuno crear una tabla que agrupe todos los detalles de las reseñas sobre la propiedad, así como una con las comodidades debido a la cantidad de variables relacionadas. En cuanto a los datos sobre la ubicación de las propiedades, se considera la normalización de la información en vista de que el área de estudio está definida para

Madrid y los barrios que la integran (todas las filas comprenden información sobre el barrio).

ANÁLISIS EXPLORATORIO DE DATOS

Se examinarán las variables y se realizarán visualizaciones para identificar patrones y relaciones entre ellas. Además, se llevará a cabo un análisis estadístico básico para comprender la distribución y el comportamiento de los datos. Esta fase de análisis exploratorio permitirá formular hipótesis y seleccionar las variables más relevantes para el modelado posterior.

Para ello, se realiza una limpieza y preparación de los datos de calidad para lograr dicho objetivo.

- Completando registros de 'Price'

Por otro lado, dado que la variable 'Price' será objeto de análisis durante todo el proyecto, se rellena valores nulos a partir de 'Weekly_price' y 'Monthly_price' para realizar un análisis más exhaustivo.

La recopilación de este dato a partir de las columnas de precio semanal y mensual, se hará dividiendo el precio semanal entre 7 días y el mensual entre 30.

En un primer comienzo, se baraja la posibilidad de que las propiedades que no tengan información en la columna 'Price' pueda deberse a que el propietario de la vivienda no quiera ofrecer el servicio a corto plazo. Sin embargo, haciendo comprobaciones previas, se encuentran incongruencias donde la propiedad carece de información de precio por noche, pero sin embargo tiene '*Minimum nights*' < 6 días. Por lo tanto, se descarta la hipótesis inicial y se da por válido el cálculo de estos registros faltantes a partir de los precios semanales y mensuales.

```
condition = (df_airbnb['PRICE'].isnull()) & ((df_airbnb['WEEKLY_PRICE'].notnull()) | (df_airbnb['MONTHLY_PRICE'].notnull()))
result = df_airbnb[condition]
result

df_airbnb['PRICE_CLEANSSED'] = np.where(df_airbnb['PRICE'].isnull(), df_airbnb['WEEKLY_PRICE_CLEANSSED'] / 7, df_airbnb['PRICE'])
df_airbnb['PRICE_CLEANSSED'] = np.where(df_airbnb['PRICE'].isnull(), df_airbnb['MONTHLY_PRICE_CLEANSSED'] / 30, df_airbnb['PRICE'])
```



```
df_filtered = df_airbnb.loc[ (df_airbnb['PRICE'].isnull()) &
                             ((df_airbnb['WEEKLY_PRICE'].notnull()) | (df_airbnb['MONTHLY_PRICE'].notnull()))
                           ]

df_filtered[['PRICE', 'WEEKLY_PRICE', 'MONTHLY_PRICE', 'MINIMUM_NIGHTS']]
```

"""Obtenemos:

	PRICE	WEEKLY_PRICE	MONTHLY_PRICE	MINIMUM_NIGHTS
4353	NaN	NaN	1843.0	1
6071	NaN	NaN	20000.0	6
6975	NaN	910.0	2910.0	5
10238	NaN	910.0	2910.0	5

- Cálculo del precio total

En este caso, se tendrá en cuenta el coste de limpieza y el coste unitario por noche para al cálculo del precio total. el depósito no se considera un coste del precio total ya que éste es recuperable.

```
#Compruebo que la columna cleaning fee no tiene Nans
df_airbnb['CLEANING_FEE'].isna().sum() # 0 NaN OK

#Realizo la operación
df_airbnb['TOTAL_PRICE'] = df_airbnb['PRICE_CLEANSED'] + df_airbnb['CLEANING_FEE']

print(df_airbnb.columns) #comprobamos que nos ha creado la columna
```

- Descuentos por alquiler media-larga temporada

Para realizar este estudio se coge una muestra de entradas y se comprueba si los precios semanales y mensuales ofrecen descuento o si son proporcionales en el tiempo.

Para llevarlo a cabo se crea un DataFrame en el que se elimina los NaN de las variables 'Price', 'Weekly_price' y 'Monthly_price' y se representan las veinte primeras entradas.

```
df_prueba_Nan = df_airbnb[['PRICE', 'WEEKLY_PRICE', 'MONTHLY_PRICE']].dropna()
df_prueba_Nan.head(20)
```

Se crean las mismas columnas calculadas analíticamente y se comprueba que el resultado converge con el anterior.

```
df_prueba_Nan['WEEKLY_PRICE_CALCULATED'] = df_prueba_Nan['PRICE'] * 7
```

Se comprueba si todas las viviendas que ofrecen un precio semanal tienen descuento.

```
df_prueba_Nan["WEEKLY_PRICE_CALCULATED"] > df_prueba_Nan["WEEKLY_PRICE"]
(df_prueba_Nan["WEEKLY_PRICE_CALCULATED"] > df_prueba_Nan["WEEKLY_PRICE"]).value_counts()
```

Esto da como resultado 2158 apartamentos que ofrecen un descuento semanal, mientras que 798 no lo ofrecen. Este resultado no es aclaratorio, por lo que se procede a comprobar los casos en el que el precio semanal es superior al $PRICE \times 7$, pues esto significa que el precio del alquiler de la vivienda semanal es muy superior que alquilarla por días sueltos.

```
df_prueba_Nan[['WEEKLY_PRICE_CALCULATED', 'WEEKLY_PRICE']]
df_prueba_Nan[df_prueba_Nan['WEEKLY_PRICE'] > df_prueba_Nan['WEEKLY_PRICE_CALCULATED']]
```

Algunos ejemplos de estos casos son:

PRICE	WEEKLY_PRICE	MONTHLY_PRICE	WEEKLY_PRICE_CALCULATED
95	750	3000	665
65	500	2400	455
20	273	1092	140

Con ello se concluye que calcular los valores de *'Weekly_price'* y *'Monthly_price'* proporcionalmente al precio diario no es lo más acertado. Rellenar los valores NAN con valores calculados de $PRICE \times 7$ o $PRICE \times 30$, dependiendo si es semanal o mensual, no es preciso, por lo que se crea una nueva columna que indique si el arrendador tiene descuentos en estos periodos para un nuevo enfoque de análisis.

Para ello se crea la columna *'Weekly_discount'* que devuelve un booleano en el caso de que *'Weekly_price'* sea menor que $PRICE \times 7$. Esto devuelve un total de 12.148 viviendas con un descuento semanal.

```
df_airbnb['WEEKLY_DISCOUNT'] = df_airbnb['WEEKLY_PRICE'] < df_airbnb['PRICE'] * 7
df_airbnb['WEEKLY_DISCOUNT'].value_counts()
```

Se realiza lo mismo con los precios mensuales. Esto devuelve un total de 3.267 viviendas con un descuento mensual.

```
df_airbnb['MONTHLY_CALCULATED'] = df_airbnb['PRICE'] * 30
df_airbnb['MONTHLY_DISCOUNT'] = df_airbnb['MONTHLY_PRICE'] < df_airbnb['MONTHLY_CALCULATED']
df_airbnb['MONTHLY_DISCOUNT'].value_counts()
```

```
df_airbnb[['HOST_RESPONSE_RATE', 'HOST_RESPONSE_TIME', 'HOST_LISTINGS_COUNT', 'HOST_TOTAL_LISTINGS_COUNT', 'CALCULATED_HOST_LISTINGS_COUNT']] = df_airbnb[
```

- **Precio por persona**

Se calcula una nueva columna de precio por persona. La ausencia de características cualitativas de la vivienda interfiere en la exactitud del análisis. Por ello, se realizará este cálculo para entender mejor el comportamiento de los precios del alojamiento.

```
df_airbnb['PRICE_PER_PERSON']=round(df_airbnb["PRICE_CLEANSED"] / df_airbnb["ACCOMMODATES"],2)
```

- **Ratio cama vs persona**

El cálculo de esta columna es trivial, sólo hay que dividir el número de camas entre el número de personas que puede acomodar la vivienda.

```
df_airbnb['RATIO_BEDS_PERS']=round(df_airbnb["BEDS"] / df_airbnb["ACCOMMODATES"] ,2)
```

- **Ocupación máxima y tasa de ocupación ('Ratio_Nvecas_alquilado')**

Esta métrica muy importante a tener en cuenta a la hora de analizar la rentabilidad de los alojamientos y constituirá uno de los principales KPI del informe. Para ello, se crea un campo que defina esta rentabilidad sabiendo el número de veces que una vivienda pudo haber sido alquilada desde que está activa en la plataforma y el número de veces que realmente fue alquilada. Los pasos que seguir (o pseudocódigo) serán:

- Calcular los días que lleva la vivienda publicada en la plataforma a partir de la diferencia entre 'last_review' y 'first_review' siguiendo las suposiciones iniciales comentadas en el apartado 'Consideraciones previas'.

```
df_airbnb['DAYS_POSTED'] = (df_airbnb['LAST_REVIEW'] - df_airbnb['FIRST_REVIEW']).dt.days #Devuelve el resultado en días
#Para evitar valores infinitos en cálculos posteriores , aseguramos que como mínimo todos los alojamientos han estado publicados por un día, para ello
#Hay casos que la first_review y last_review son el mismo día. Esto hace que la diferencia sea 0.
df_airbnb['DAYS_POSTED'] = df_airbnb['DAYS_POSTED'].replace(0, 1)
```

- Cálculo del máximo nivel de ocupación dividiendo el número de días publicado entre el número de noches mínimas establecidas por el 'Host'. Por ejemplo, si el alojamiento lleva publicado 30 días y el número de noches mínimas que se puede alquilar son 10, el máximo número de veces que el alojamiento se puede alquilar serían 3 veces.

```
df_airbnb['MAX_OCCUPANCY'] = df_airbnb['DAYS_POSTED'] / df_airbnb['MINIMUM_NIGHTS']
```

- Calcular la ratio de número de veces alquilado. Esto representará las veces que el alojamiento fue alquilado respecto a las que podría haber sido alquilado. Siguiendo con el ejemplo anterior, si el máximo de ocupación del alojamiento es de 3 veces, pero realmente estuvo ocupado unas 2 veces, la ratio será del 66.6%.

```
df_airbnb['RATIO_NVECES_ALQUILADO'] = df_airbnb['NUMBER_OF_REVIEWS'] / df_airbnb['MAX_OCCUPANCY']
df_airbnb[['RATIO_NVECES_ALQUILADO', 'NUMBER_OF_REVIEWS', 'MAX_OCCUPANCY', 'DAYS_POSTED']]
```

- Ratio cama vs huéspedes

El cálculo consiste en dividir el número de habitaciones entre el número de huéspedes que es capaz de contener la vivienda. En esta columna hay casos especiales que resultan valores infinitos, como el caso en el que la vivienda no cuenta con dormitorio.

```
df_airbnb['RATIO_ROOMS_PERS'] = round(df_airbnb['BEDROOMS'] / df_airbnb['ACCOMMODATES'], 2)
```

- Limpieza de localización

De todos los datos que dan información sobre la ubicación de las viviendas se comprueba que la columna de localización es la más efectiva para filtrar el DataFrame por Madrid, puesto que tiene menos NAN e identifica mejor Madrid.

- o *State*: tiene 13.205 resultados en Madrid.

```
filtered_state = df_airbnb[df_airbnb['STATE'].str.contains("MADR|MAD", case=False)]
count_state = filtered_state['STATE'].count()
print('Hay', count_state, 'resultados para State conteniendo Madr o Mad') #13205
```

- o *City*: tiene 13.241 resultados en Madrid.

```
filtered_city = df_airbnb[df_airbnb['CITY'].str.contains("MADR|MAD", case=False)]
count_city = filtered_city['CITY'].count()
print('Hay', count_city, 'resultados para City conteniendo Madr o Mad') #13241
```

- o *Smart location*: 13.243 resultados en Madrid.

```
filtered_smart_loc = df_airbnb[df_airbnb['SMART_LOCATION'].str.contains("MADR|MAD", case=False)]
count_smart_loc = filtered_smart_loc['SMART_LOCATION'].count()
print('Hay', count_smart_loc, 'resultados para Smart Location conteniendo Madr o Mad') #13243
```

- o *Market*: tiene 13.268 resultados en Madrid.

```
filtered_market = df_airbnb[df_airbnb['MARKET'].str.contains("MADR|MAD", case=False)]
count_market = filtered_market['MARKET'].count()
print('Hay', count_market, 'resultados para Market conteniendo Madr o Mad') #13268
```

- o *Latitude*: para este caso hay que tener en cuenta que todas las latitudes de Madrid empiezan por 40. Esta es la mejor opción, presenta 13.399 registros y también recoge las ciudades de Madrid que no contienen el *string* de 'Madrid' o parte de él. Como por ejemplo 'Aravaca'.

```
filtered_latitude = df_airbnb[(df_airbnb['LATITUDE'] >= 40.0) & (df_airbnb['LATITUDE'] <= 41.0) ]
count_latitude = filtered_latitude['LATITUDE'].count()
print('Hay', count_latitude, 'resultados para Latitud empezando por 40') #13399 --> La mejor forma de identificar Madrid
```

```
df_airbnb[(df_airbnb['LATITUDE'] >= 40.0) & (df_airbnb['LATITUDE'] <= 41.0) & ~(df_airbnb['CITY'].str.contains("MADR|MAD", case=False))]
```

Se crea una nueva columna, *'City_cleansed'*, que sea Madrid para todas las latitudes 40 con *'Country'* igual a *Spain*, con lo que se obtiene un dato limpio de localización.

```
df_airbnb['CITY_CLEANSSED'] = np.where((df_airbnb['LATITUDE'] >= 40.0) & (df_airbnb['LATITUDE'] <= 41.0) & (df_airbnb['COUNTRY'] == 'SPAIN'), 'MADRID',
```

- División y factorización de *'Amenities'*

Se crean columnas para las amenities más relevantes, de forma que si la propiedad la ofrece devuelve un booleano. Las amenities consideradas en este trabajo son:

- Internet

```
df_airbnb["INTERNET"] = df_airbnb["AMENITIES"].str.contains("INTERNET")
```

- Aire acondicionado

```
df_airbnb["AC"] = df_airbnb["AMENITIES"].str.contains("AIR CONDITIONING")
```

- Checking en 24 horas

```
df_airbnb["24H_CHECKING"] = df_airbnb["AMENITIES"].str.contains("24-HOUR CHECK-IN")
```

- Limpieza

```
df_airbnb["WASHER"] = df_airbnb["AMENITIES"].str.contains("WASHER")
```

- Fumar en la propiedad

```
df_airbnb["SMOKING ALLOWED"] = df_airbnb["AMENITIES"].str.contains("SMOKING ALLOWED")
```

- Acepta mascotas

```
df_airbnb["PETS ALLOWED"] = df_airbnb["AMENITIES"].str.contains("PETS ALLOWED")
```

- Ascensor

```
df_airbnb["ELEVATOR IN BUILDING"] = df_airbnb["AMENITIES"].str.contains("ELEVATOR IN BUILDING")
```

- Columna booleana de 'security deposit'

Esta columna nueva devuelve un booleano que indica que si hay que dejar una fianza al entrar a la vivienda vacacional.

Primero se debe comprobar si 'Security_deposit' contiene NAN.

```
df_airbnb['SECURITY_DEPOSIT'].isna().sum() #0 ok
```

Finalmente se crea la columna que devuelve el valor booleano.

```
df_airbnb['SECURITY_DEPOSIT_BOOL'] = df_airbnb['SECURITY_DEPOSIT'] > 0
```

- Codificando políticas de cancelación

Con la intención de simplificar el análisis posterior sobre las políticas de cancelación, se normalizan los valores únicos (*moderate*, *strict*, *flexible*, *super_strict_60*, *super_strict_30*, *flexible_new*, *moderate_new*, *strict_new*). en este caso, *flexible_new*, *moderate_new* y *strict_new* se convertirán a flexible, moderate y strict respectivamente, ya que no se considera ninguna diferencia entre esos valores. Por otro lado, se codificará del 1 al 5 del menos estricto al más estricto los tipos de cancelación.

```
df_airbnb['CANCELLATION_POLICY'].replace({"FLEXIBLE_NEW": "FLEXIBLE",  
                                           "MODERATE_NEW": "MODERATE",  
                                           "STRICT_NEW": "STRICT"}, inplace=True)
```

Se comprueba que los datos son correctos tras esta operación.

```
df_airbnb['CANCELLATION_POLICY'].unique() #ok
```

Luego se hace el recuento para comprobar que la codificación es satisfactoria.

```
df_airbnb['CANCELLATION_POLICY'].value_counts()  
  
#      STRICT      5773  
#    FLEXIBLE    4680  
#    MODERATE    4263  
#  SUPER_STRICT_60     32  
#  SUPER_STRICT_30     19
```

Se codifican los 5 niveles mencionados anteriormente.

```
# crear el diccionario de mapeo  
map_dict = { 'FLEXIBLE': 1, 'MODERATE': 2, 'STRICT': 3, 'SUPER_STRICT_30': 4, 'SUPER_STRICT_60': 5}  
  
# aplicar el método map en la columna "CANCELLATION_POLICY"  
df_airbnb['CANCELLATION_POLICY'] = df_airbnb['CANCELLATION_POLICY'].map(map_dict)  
  
#Comprobamos recuento  
  
df_airbnb['CANCELLATION_POLICY'].value_counts()  
  
#3    5773  
#1    4680  
#2    4263  
#5     32  
#4     19    Correcto
```

- Ratio del número de veces alquilado

Primero se comprueba que esta ratio pueda tener valores mayores que la unidad.

```
df_airbnb.sort_values("RATIO_NVECES_ALQUILADO", ascending=False).head(20)[["ID", "NUMBER_OF_REVIEWS", "MAX_OCCUPANCY", "MINIMUM_NIGHTS", "DAYS_POSTED",
```

Luego, se filtran los casos por ID definidos para ver la explicación más detenidamente de porqué se obtienen estos resultados.

```
filtered_df = df_airbnb[df_airbnb['ID'].isin([844856, 755184, 4532131])]  
filtered_df[["FIRST_REVIEW", "LAST_REVIEW", "MINIMUM_NIGHTS", "DAYS_POSTED", "NUMBER_OF_REVIEWS"]]
```

Con esta representación se puede confirmar que hay alojamientos que tienen un mínimo de noches para alquilar que no se han cumplido. Se considera que esto se deba a que posiblemente los 'Host' hicieron las modificaciones del periodo mínimo del alquiler del alojamiento a posteriori. Es decir, es el caso del alojamiento que tiene un mínimo de 60 noches y se ha alquilado en 2 ocasiones sin estar publicado más de 15 días.

Finalmente se comprueban los tipos de datos.

```
df_tipo_datos = df_airbnb.dtypes.to_frame().reset_index()  
df_tipo_datos.columns = ['Nombre columna', 'Tipo dato']  
df_tipo_datos.head(30)
```

- Análisis de la ratio de ocupación > 1

Realizando un análisis sobre la nueva ratio calculada se encuentran casos con este valor por encima de uno, para ello se realiza un análisis más detallado.

Primero se saca el recuento de los casos con 'Ratio_Nveces_alquilado' mayor que 1 en df_airbnb.

```
mask = df_airbnb["RATIO_NVECES_ALQUILADO"] > 1  
count = df_airbnb[mask].shape[0]  
print("Hay", count, "registros con RATIO_NVECES_ALQUILADO mayor a 1")
```

Hay un total de 911 casos con esta peculiaridad que se debe estudiar. Se analizan unos pocos registros para mayor entendimiento.

```
df_airbnb.loc[df_airbnb["RATIO_NVECES_ALQUILADO"] > 1, ["ID", "DAYS_POSTED", "MINIMUM_NIGHTS", "NUMBER_OF_REVIEWS", "RATIO_NVECES_ALQUILADO"]]
```

Mediante esta representación del código, se puede observar que esto se debe a que no concuerdan los días que el alojamiento lleva publicado, y las veces máximas que se ha podido alquilar. Es decir, casos en los que muestra que la propiedad ha sido alquilada más veces, en función de N veces alquilado = número de reviews, de las que puede.

Por ejemplo, el alojamiento lleva publicado 20 días y tiene un *'Minimum_nights'* de 10 noches y tiene 4 reviews. En este caso, el máximo de ocupación sería:

$$\text{max_occupancy} = 20 \text{ días} / 10 \text{ minimum nights}$$

Como máximo la propiedad se puede alquilar 2 veces y, sin embargo, tiene 4 reviews. Esto indica que se ha alquilado 4 veces. Dos veces por encima de sus posibilidades, esto resultaría en una ratio de 2 o 200%. Una explicación para ello, sería que el host inicialmente tuviese establecido ciertas noches mínimas, y las modificase posterior a algunos de los alojamientos.

Otros motivos de resultados extraños en la ratio, son propiedades con valores de 0 en días publicados y/o *'minimum_nights'*. Una forma de solventarlo, asumir que todas las propiedades que, se hayan alquilado 1 vez, al menos estén 1 día publicados. Así también modificamos posibles valores infinitos.

Como conclusión, a todas las viviendas que tengan 1 *DAY_POSTED* y 1 REVIEW, tendrán el *RATIO_NVECES_ALQUILADO = 100%*.

```
df_airbnb.loc[(df_airbnb['DAYS_POSTED'] == 1) & (df_airbnb['NUMBER_OF_REVIEWS'] == 1), 'RATIO_NVECES_ALQUILADO'] = 1
df_airbnb.loc[df_airbnb['RATIO_NVECES_ALQUILADO'] > 1, ['ID', 'DAYS_POSTED', 'MINIMUM_NIGHTS', 'NUMBER_OF_REVIEWS', 'RATIO_NVECES_ALQUILADO']]
```

El resto de los valores que sobrepasen la unidad se consideran outliers, ya que no sería preciso manipular sus datos.

Con esta modificación se consigue reducir a 136 registros con la ratio mayor que 1, aproximadamente un 1% de los datos totales.

- Exportar el nuevo CSV

Se exporta el df con *'City_cleansed' = 'MADRID'* en formato csv para poder tratarlo en Dveaber y, posteriormente, en Tableau y R.

```
df_airbnb_cleansed = df_airbnb[df_airbnb['CITY_CLEANSSED'] == 'MADRID']
df_airbnb_cleansed.to_csv('df_airbnb_cleansed.csv', index=False, sep=',')
```


PROCESO EN POSTGRESQL – SISTEMA DE GESTIÓN USADA PARA IMPLEMENTAR LA BASE DE DATOS

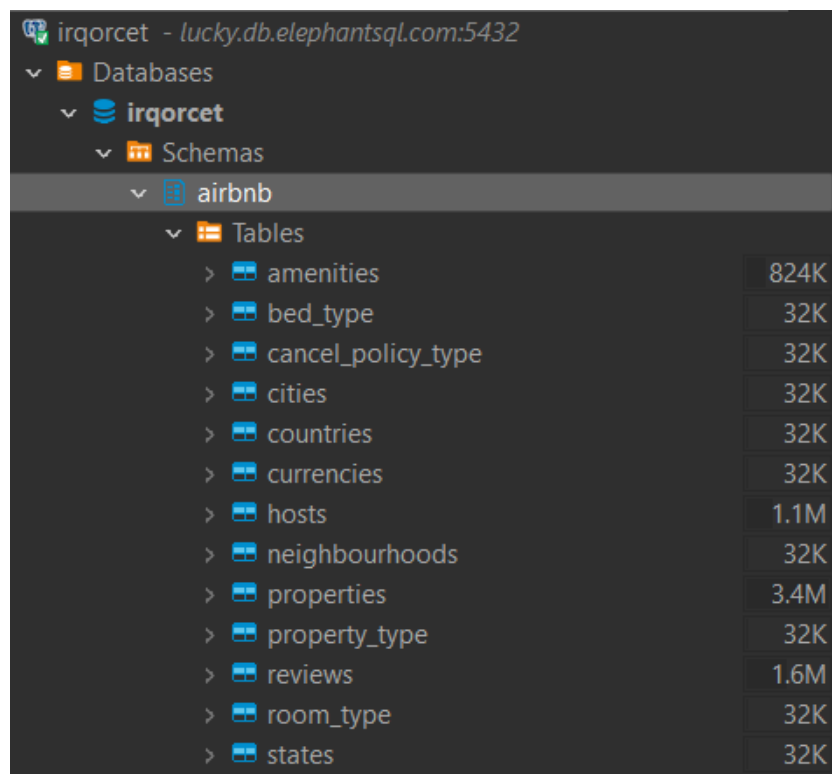
Siguiendo el esquema fijado en el diagrama E-R y luego de haber limpiado, normalizado y exportado la información relevante para el proyecto, mediante un script de SQL (DDL) se crearon las tablas correspondientes, se definió la estructura, los tipos de datos y se cargó parte de la información (la otra parte se importó de los CSV hechos en Python):

```
--Tabla de los tipos de propiedades
create table airbnb.property_type(
  ID_PROPERTY_TYPE varchar(100) not null, --PK
  NAME varchar(50) not null,
  DESCRIPTION varchar(512) null
);

alter table airbnb.property_type
add constraint property_type_PK primary key (ID_PROPERTY_TYPE);

-- Carga de datos
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('1', 'APARTMENT', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('2', 'BED & BREAKFAST', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('3', 'BOAT', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('4', 'BOUTIQUE HOTEL', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('5', 'BUNGALOW', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('6', 'CAMPER/RV', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('7', 'CASA PARTICULAR', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('8', 'CHALET', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('9', 'CONDOMINIUM', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('10', 'DORM', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('11', 'EARTH HOUSE', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('12', 'GUEST SUITE', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('13', 'GUESTHOUSE', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('14', 'HOSTEL', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('15', 'HOUSE', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('16', 'LOFT', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('17', 'OTHER', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('18', 'SERVICED APARTMENT', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('19', 'TENT', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('20', 'TIMESHARE', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('21', 'TOWNHOUSE', ' ');
insert into airbnb.property_type (ID_PROPERTY_TYPE, NAME, DESCRIPTION) values('22', 'VILLA', ' ');
```

Figura 3: Porción del DDL Script



irqorcet - lucky.db.elephantsql.com:5432	
Databases	
irqorcet	
Schemas	
airbnb	
Tables	
> amenities	824K
> bed_type	32K
> cancel_policy_type	32K
> cities	32K
> countries	32K
> currencies	32K
> hosts	1.1M
> neighbourhoods	32K
> properties	3.4M
> property_type	32K
> reviews	1.6M
> room_type	32K
> states	32K

Figura 4: Base de Datos – tablas creadas con información cargada

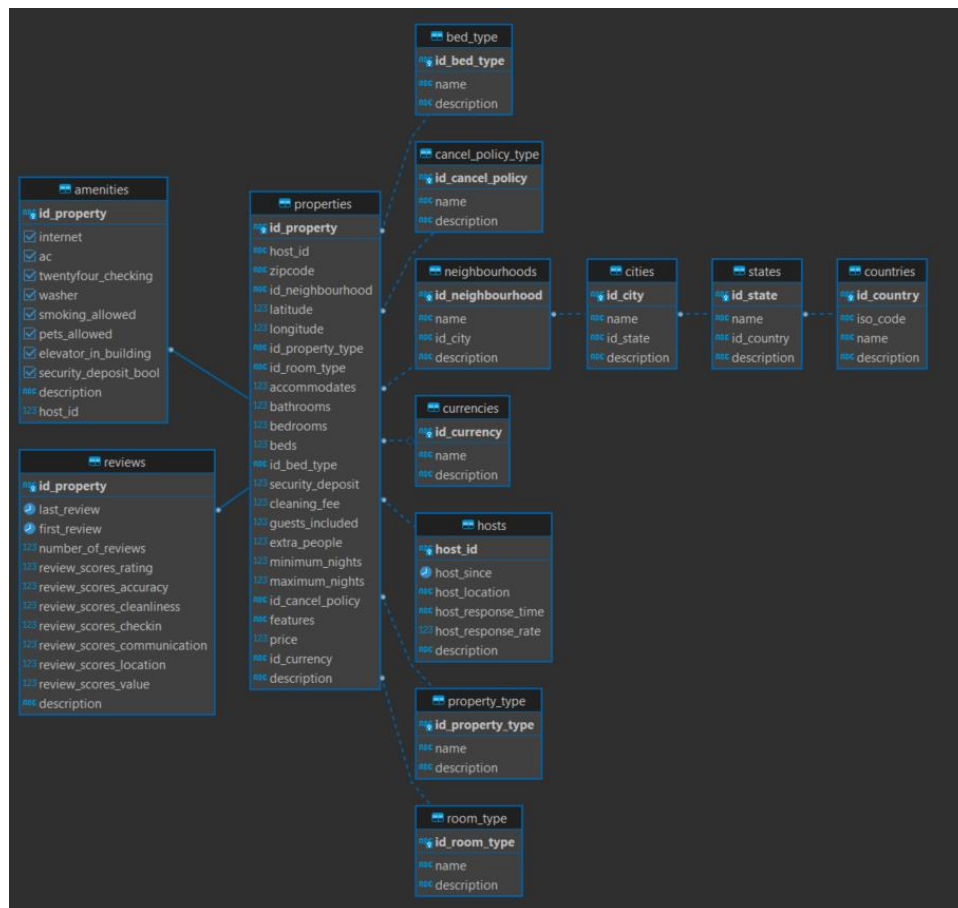


Figura 5: Diagrama E-R en PostgreSQL

VISUALIZACIÓN DE LAS MÉTRICAS

En este apartado se expondrá el cuadro de mandos realizado en la plataforma Tableau para el análisis de KPIS relevantes de cada vivienda y/o barrio, así como la detección de acciones para maximizar la rentabilidad de dichos alojamientos de los clientes de Airbnb.

Para procesar la fuente de datos se importó el CSV generado en Jupyter Notebooks. Para que dicho procesamiento fuese correcto se cambió el delimitador de Tableau a “;”, así como la localización a Reino Unido para poder interpretar correctamente los separadores decimales de los números tipo float.

También se hizo un análisis previo para comprobar que el tipo de datos proveniente del archivo procesado por Python eran correctamente detectados por Tableau y se modificó el tipo de datos en la vista Data Source.

Posteriormente, se aplicó un filtro general de ratio de alquiler entre 0 y 1 para eliminar los outliers, puesto que aquellas viviendas con una ratio mayor que la unidad contaban con datos erróneos o en reviews o en el número de noches mínimas y no se disponía de la información suficiente para inferirlos.

Para la visualización de las métricas se ha utilizado el mencionado anteriormente programa mediante el cual se ha realizado un dashboard o cuadro de mandos que compuesto de tres módulos:

- Módulo superior: muestra una vista detallada de la vivienda de alquiler y sus KPIS.
- Módulo central: Mapa interactivo de ID y Barrio
- Módulo inferior: Gráficos de correlación de la ratio de alquiler respecto de las variables relevantes.

El objetivo del dashboard, como se menciona en el apartado ‘idea del proyecto’, es la simulación de una plataforma para el departamento dedicado a la optimización del negocio, es decir, del Airbnb. Para ello se diseña un cuadro de mandos que muestre la ratio de alquiler o el porcentaje de tiempo alquilado de cada vivienda además de otras métricas o KPIS para proporcionar al propietario de dicho alojamiento medidas de mejora para optimizar la ratio y, por lo tanto, aumentar los beneficios.

Por lo tanto, la variable principal o KPI del dashboard es la ratio de alquiler, previamente explicada su obtención. Este valor se compara con la ratio de alquiler medio del barrio al que pertenece la vivienda y con el de la Comunidad de Madrid. También se hace uso de métricas como precio por persona y noche, coste de la fianza, coste de la limpieza, número de habitaciones, número de camas, número de huéspedes y puntuaciones de las reviews referidas a limpieza, comunicación, localización y veracidad del listing en Airbnb.

- **Módulo superior**

Para su creación se emplearon principalmente worksheets con etiquetas de texto para visualizar los diferentes KPIs de la vivienda con respecto a su barrio y a la Comunidad de Madrid (*Figura 6*).

Para el cálculo de las etiquetas por barrio y Comunidad de Madrid se hace uso de herramientas intermedias, que son campos calculados que permiten agregar los datos por grupos. Para realizar esta tarea se emplea el comando FIXED, de esta manera se pueden excluir los filtros de dimensión (filtro ID) y agregar por grupos. Un ejemplo de esta última aplicación sería:

`{FIXED([NEIGHBOURHOOD_CLEANSED]):AVG([RATIO_NVECES_ALQUILADO])}`.

También se incorpora un gráfico circular para mostrar de una manera más visual el rendimiento de la vivienda en función de su ratio de alquiler. Este gráfico cambia en función del ID o de la vivienda mostrando el rendimiento de ella. Un rendimiento del 100% equivale a un círculo completo. Para su construcción se han unido dos gráficos circulares superpuestos sobre el mismo eje. Posteriormente, se modificó su diseño para darle ese aspecto de circunferencia dinámica y se añadió la etiqueta de texto en el centro para mostrar el progreso.



Figura 6: Módulo Superior

En este módulo también hay dos tablas que muestran diferentes KPI's de la vivienda respecto al barrio y la Comunidad de Madrid por medio de etiquetas y campos agrupados calculados. También se desarrolló un filtro “semáforo” para poder filtrar los IDs del mapa en función de su ratio de alquiler. Este “semáforo” se construye teniendo en cuenta que LOW serán las viviendas con una ratio de alquiler inferior al 33%, MEDIUM será para aquellas con una ratio entre el 33% y el 66% y HIGH será para aquellas con una ratio superior al 66%. Para obtenerlo se comienza creando un campo controlado por IF y ELSE que clasifiquen los ID's y se presenta en forma de gráfico circular por colores. Se habilitó la opción de filtro del dashboard para afectar a todas las etiquetas y gráficos de los módulos.

- **Módulo central**

Se trata de un mapa interactivo (Figura 7). Para su creación se emplean los campos LATITUDE y LONGITUDE y los ID. Se emplea un código de color indicando con un tono más verde los ID's que incluyen las viviendas con mayor ratio de alquiler y los rojos los menores. La descripción emergente (Figura 7) muestra los datos más relevantes del alquiler, como es el ID, el barrio, los huéspedes, el precio por persona, precio del alquiler por noche y otros servicios que ofrece el alojamiento. Los servicios se representan con checks verdes o cruces rojas sustituyendo el True/False original del data set mediante la utilización de alias.

Este mapa, además de ser interactivo, sirve de filtro de acción para el módulo 1, pudiendo visualizar los datos de la vivienda al pinchar en él. Sin embargo, el módulo 3 quedaría exento de dicho filtro.

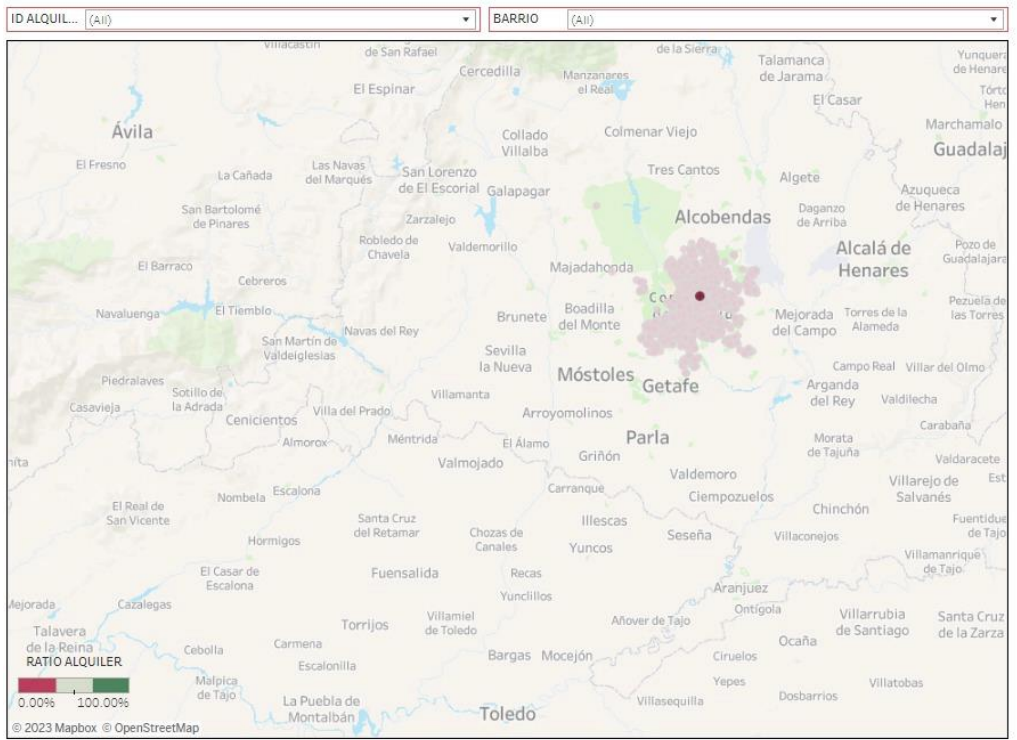


Figura 7: Módulo central

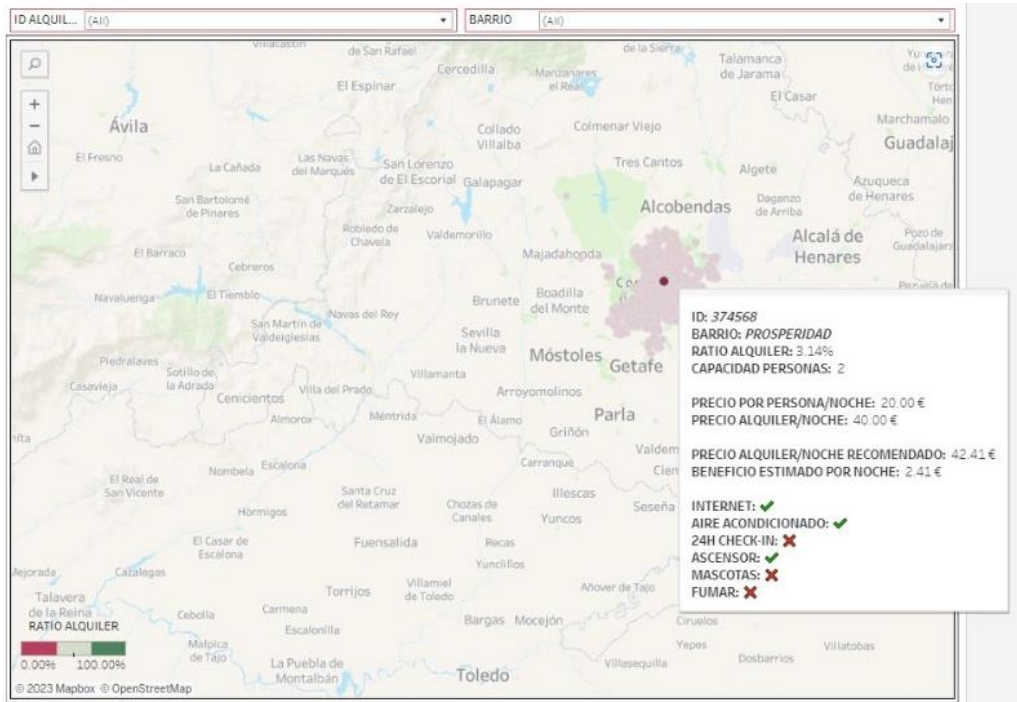


Figura 8: Módulo central con ventana emergente

- **Módulo inferior**

Este módulo muestra diferentes gráficos de correlación (*Figura 9*) que sirven para argumentar los cambios necesarios para mejorar la ratio de alquiler de cada vivienda. Estos gráficos dependen del filtro de barrio o del “semáforo” de la ratio de alquiler, pero no dependen del ID, puesto que su objetivo es demostrar la correlación genérica entre las viviendas del barrio o del total de viviendas de Madrid.

En ellos se puede observar la relación entre el precio por persona, el número de habitaciones, el número de camas, el número de huéspedes, el coste de fianza, las el coste de limpieza, las reviews de comunicación y las reviews de limpieza con la ratio de alquiler.

Con la ayuda de estos gráficos se puede observar, por ejemplo,

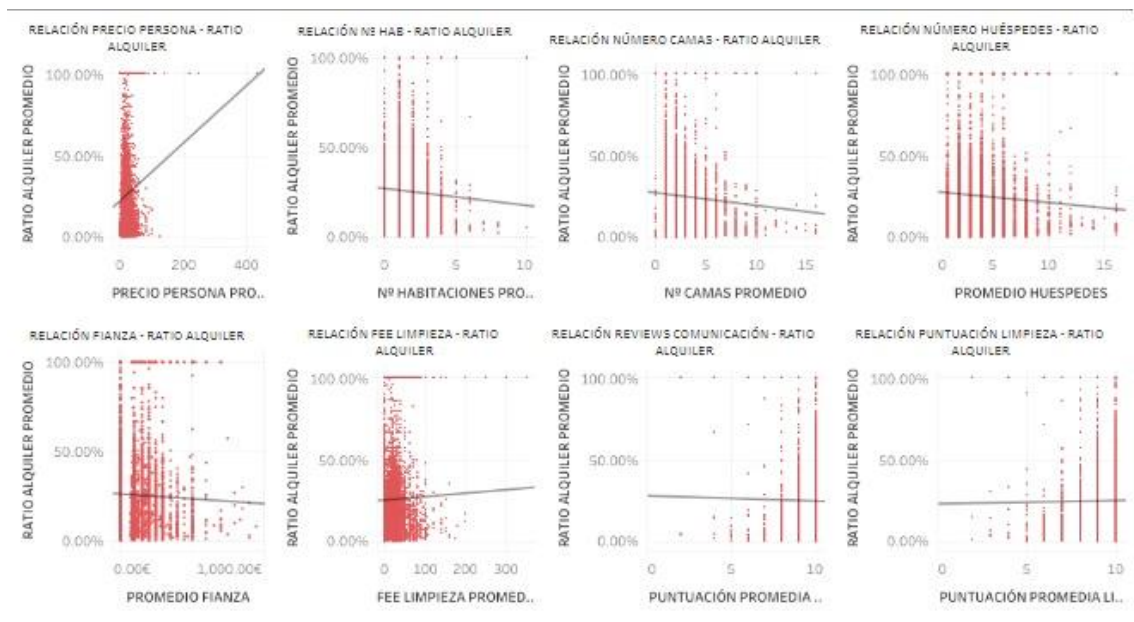


Figura 9: Módulo inferior

Mediante estas gráficas se observa como la correlación de las diferentes variables varía según la agregación de la ratio de alquiler de las viviendas. Siendo aquellas con una ratio “HIGH” menos susceptibles a dichas variables que aquellas que tienen una ratio “LOW”.

Por ejemplo, se puede observar que aquellas casas con una ratio de alquiler entre 0% y 33% (“LOW”) tienen un impacto mucho más positivo a la ratio de alquiler a mejor puntuación en las reviews de Limpieza y Comunicación, mientras que las viviendas con Ratio “HIGH”, apenas aprecian cambios en su ratio de alquiler por estas variables, hecho que puede explicarse por el ya alto ratio de alquiler con el que cuentan dichas viviendas.

También es interesante observar, que aquellas casas clasificadas bajo la ratio “LOW” son mucho más sensibles al precio por persona que las casas con altos ratios de alquiler, puesto que posiblemente cuando una vivienda cuenta con poca tasa de alquiler sea porque hay otras variables afectando a la demanda y un mayor precio perjudique aún más su demanda.

Además, se observan diferencias según el en el barrio en el que se encuentran siendo posible extraer conclusiones y recomendaciones a los propietarios en función de donde se sitúe su vivienda.

PROCESAMIENTO Y MODELADO

- Importar .csv

Se importa el .csv de trabajo una vez revisados y normalizados los datos.

```
library(readr)
df_airbnb_cleansed <- read_csv("Documents/Bootcamp/PROYECTO FINAL/df_airbnb_cleansed.csv",
na = "NA", show_col_types = FALSE)
```

- Resumen DataFrame

ID	HOST_ID	HOST_SINCE	HOST_LOCATION
Min. : 18628	Min. : 17453	Min. : 2009-05-17	Length:13327
1st Qu.: 5859822	1st Qu.: 7823688	1st Qu.: 2013-07-31	Class :character
Median :11578007	Median : 27526991	Median : 2015-02-10	Mode :character
Mean :10439903	Mean : 37752322	Mean : 2014-10-19	
3rd Qu.:15384892	3rd Qu.: 57983871	3rd Qu.: 2016-02-10	
Max. :18109842	Max. :124753355	Max. : 2017-04-07	
	NA's :3		
HOST_RESPONSE_TIME	HOST_RESPONSE_RATE	HOST_LISTINGS_COUNT	HOST_TOTAL_LISTINGS_COUNT
Length:13327	Min. : 0.00	Min. : 0.000	Min. : 0.000
Class :character	1st Qu.: 90.00	1st Qu.: 1.000	1st Qu.: 1.000
Mode :character	Median :100.00	Median : 2.000	Median : 2.000
	Mean : 82.63	Mean : 9.741	Mean : 9.741
	3rd Qu.:100.00	3rd Qu.: 5.000	3rd Qu.: 5.000
	Max. :100.00	Max. :265.000	Max. :265.000
FIRST_REVIEW	LAST_REVIEW	NEIGHBOURHOOD_CLEANSSED	CITY
Min. :2010-05-10	Min. :2012-08-04	Length:13327	Length:13327
1st Qu.:2015-05-12	1st Qu.:2016-12-11	Class :character	Class :character
Median :2016-03-26	Median :2017-03-12	Mode :character	Mode :character
Mean :2015-11-20	Mean :2016-12-28		
3rd Qu.:2016-10-04	3rd Qu.:2017-03-29		
Max. :2017-04-07	Max. :2017-04-07		
NA's :2788	NA's :2789		
STATE	ZIPCODE	MARKET	SMART_LOCATION
Length:13327	Min. : 0	Length:13327	Length:13327
Class :character	1st Qu.: 28005	Class :character	Class :character
Mode :character	Median : 28012	Mode :character	Mode :character
	Mean : 27086		
	3rd Qu.: 28016		
	Max. :280013		
COUNTRY_CODE	COUNTRY	LATITUDE	LONGITUDE
Length:13327	Length:13327	Min. :40.33	Min. : -3.864
Class :character	Class :character	1st Qu.:40.41	1st Qu.: -3.708
Mode :character	Mode :character	Median :40.42	Median : -3.702
		Mean :40.42	Mean : -3.697
		3rd Qu.:40.43	3rd Qu.: -3.694
		Max. :40.56	Max. : -3.527

Se ve un resumen del DataFrame para ver las columnas que contiene y posibles errores residuales de la limpieza (NA, ...).

- Creación de DataFrame de trabajo

Se selecciona solo las columnas con las que se van a trabajar.

```
library(dplyr)
df_airbnb_work <- dplyr::select(df_airbnb_cleansed, "ID", "HOST_ID", "NEIGHBOURHOOD_CLEANSSED", "ZIPCODE", "LATITUDE", "LONGITUDE", "PROPERTY_TYPE", "ROOM_TYPE", "ACCOMMODATE", "BATHROOMS", "BEDROOMS", "BEDS", "BED_TYPE", "PRICE", "SECURITY_DEPOSIT", "CLEANING_FEE", "MINIMUM_NIGHTS", "NUMBER_OF_REVIEWS", "REVIEW_SCORES_RATING", "REVIEW_SCORES_ACCURACY", "REVIEW_SCORES_CLEANLINESS", "REVIEW_SCORES_CHECKIN", "REVIEW_SCORES_COMMUNICATION", "REVIEW_SCORES_LOCATION", "REVIEW_SCORES_VALUE", "CANCELLATION_POLICY", "CALCULATED_HOST_LISTINGS_COUNT", "REVIEWS_PER_MONTH", "PRICE_PER_PERSON", "RATIO_BEDS_PERS", "RATIO_ROOMS_PERS", "RATIO_NVECES_ALQUILADO", "SECURITY_DEPOSIT_BOOL")
```

El DataFrame contiene un campo de host que resulte interesante analizar para conocer si influye en el precio de una vivienda que la gestione un profesional o un particular

Se agrupa por host y se ve que hay hospedadores que tienen varias propiedades en alquiler, por lo que se supone que son gestores profesionales.

```
library(dplyr)
hosts <- df_airbnb_work |> group_by(HOST_ID) |> summarise(avg_price=mean(PRICE),
  total_houses= sum(CALCULATED_HOST_LISTINGS_COUNT), madrid_houses = n())
hosts
```

...

A tibble: 8,180 x 4

HOST_ID <dbl>	avg_price <dbl>	total_houses <dbl>	madrid_houses <int>
17453	90.00000	1	1
19854	74.00000	9	3
31622	23.00000	1	1
53526	83.33333	9	3
70273	50.00000	1	1
70604	30.00000	4	2
71597	49.00000	1	1
74966	80.00000	1	1
75744	35.00000	1	1
75944	250.00000	16	4

1-10 of 8,180 rows

Previous 1 2 3 4 5 6 ... 100 Next

```
df_airbnb_work$PRO_HOST <- (df_airbnb_work$CALCULATED_HOST_LISTINGS_COUNT > 3) == TRUE
```

Se crea una nueva variable para identificar los hospedadores profesionales. Se define que un profesional tiene anunciadas 4 viviendas o más.

Nos interesa conocer el precio de las viviendas y las posibles variables que influyen en el precio.

```
# Veo algún estimador estadístico.  
  
num_viviendas <- nrow(df_airbnb_work)  
avg_price <- mean(df_airbnb_work$PRICE)  
avg_unit_price <- mean(df_airbnb_work$PRICE_PER_PERSON)  
  
paste("El número total de viviendas es:", num_viviendas)  
paste("El precio medio diario de todas las viviendas es:", avg_price)  
paste("El precio medio diario por persona de todas las viviendas es:", avg_unit_price)
```

[1] "El número total de viviendas es: 13324"
[1] "El precio medio diario de todas las viviendas es: 67.4824377063945"
[1] "El precio medio diario por persona de todas las viviendas es: 23.20777544281"

Hay un total de 13.327 propiedades con las que trabajar con un precio medio por día de 67,48€ y un precio medio por persona de 23,21€.

- **Distribución de precios**

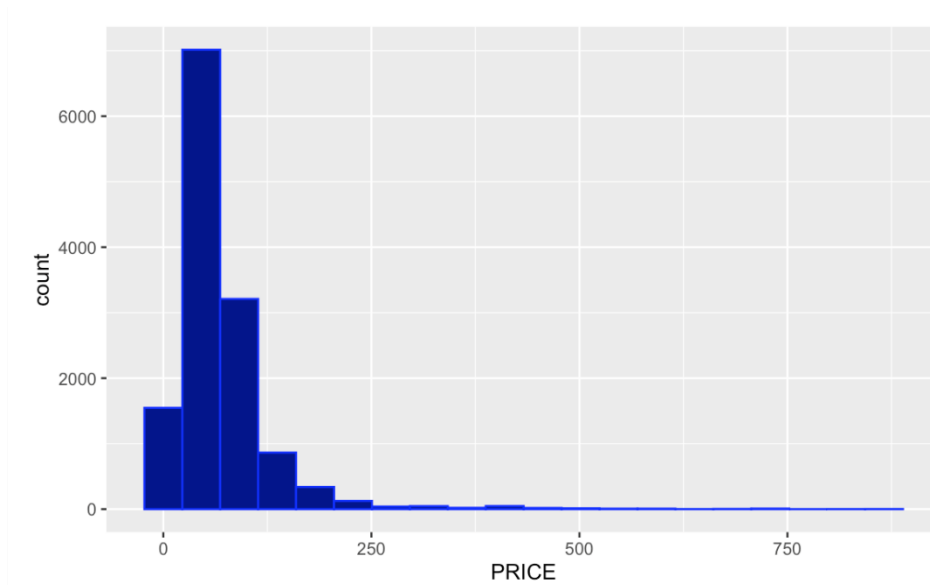


Figura 10: Distribución de precios

Se ve que la mayoría de las propiedades tienen un precio inferior a 150€, pero también hay propiedades cuyos precios son muy superiores que pueden llegar a un máximo de 875€ como se puede ver también en el resumen del DataFrame.

Se puede intuir que hay outliers, sin embargo, debido a que el precio puede depender del número de personas o habitaciones, decidimos revisar el precio unitario para detectar y eliminar outliers.

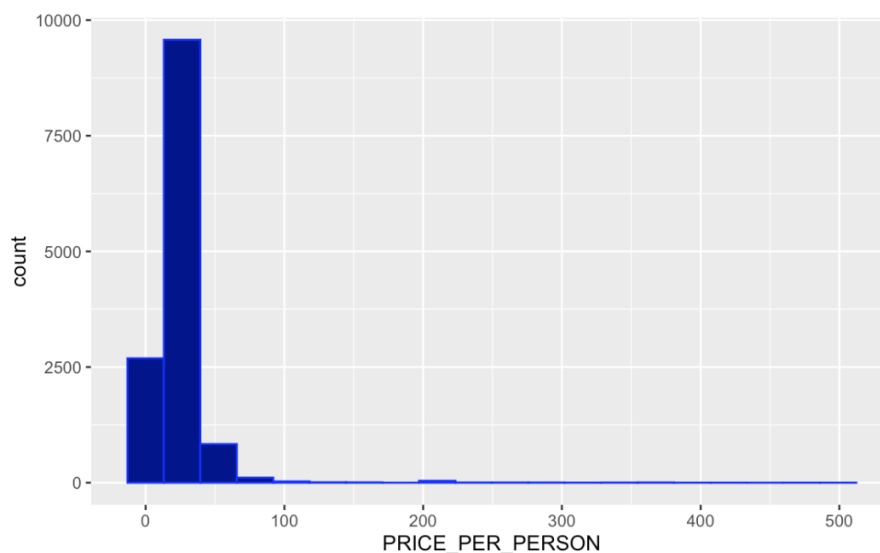


Figura 11: Distribución de precios por persona

Se ve que en el precio por persona también puede haber outliers, por lo que se analiza con un boxplot.

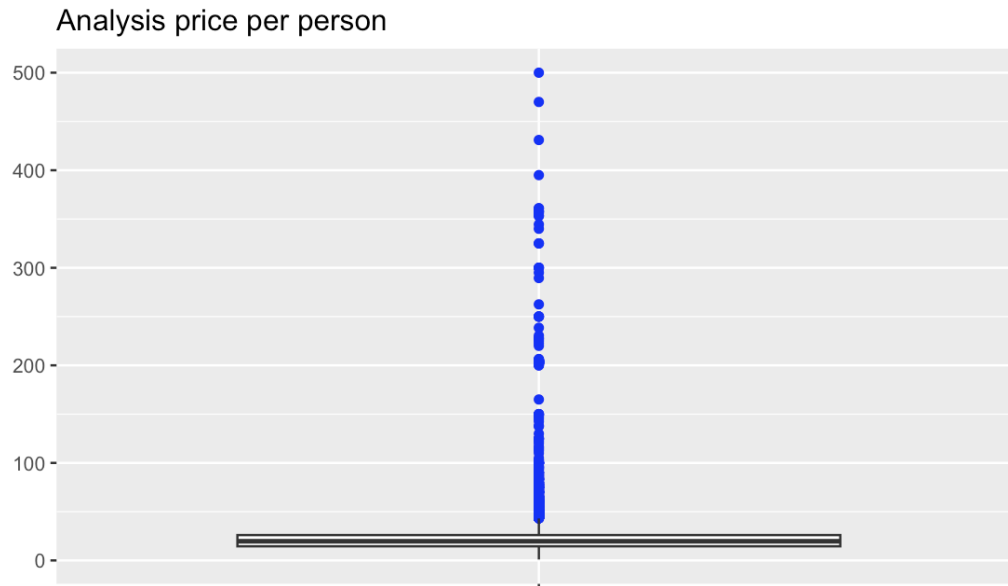


Figura 12: Análisis de precio por persona

Se detecta que hay outliers por encima de 40€ por persona.

Se eliminan del DataFrame y se confirma que se trabaja con una base sin outliers.

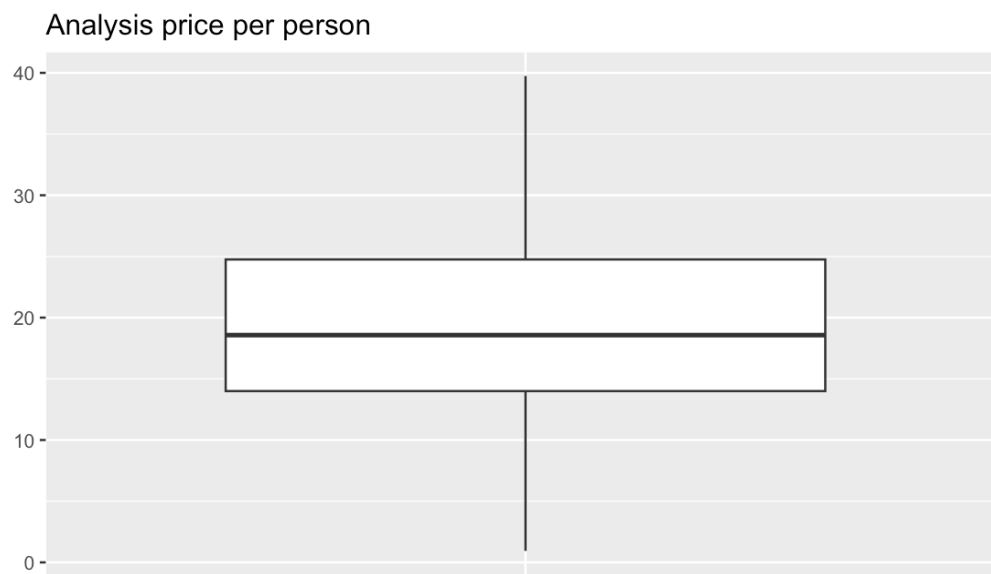


Figura 13: Análisis de precio por persona

- **Análisis por tipo de hospedador**

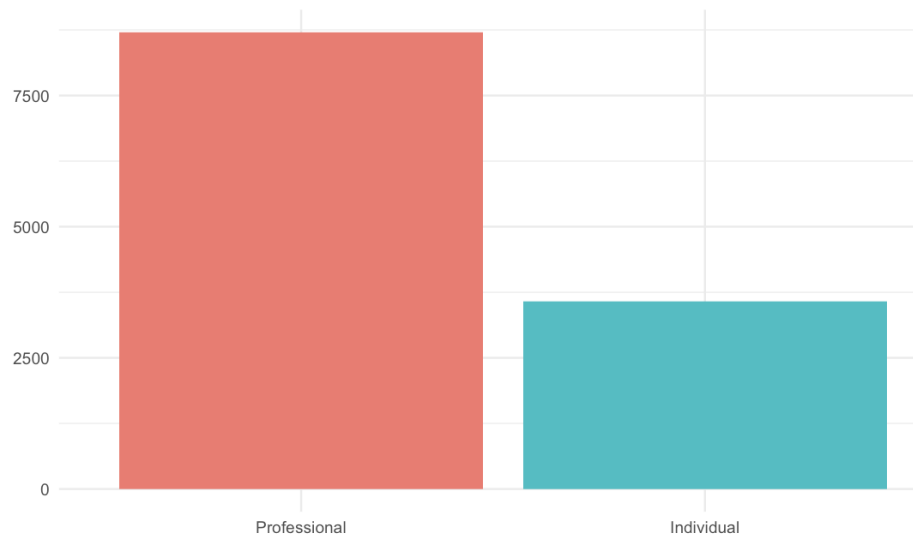


Figura 14: Análisis del precio por tipo de hospedador

Se ve que hay más propiedades gestionadas por profesionales que por particulares.

Después, se analizó el precio medio de las viviendas por tipo de hospedador.

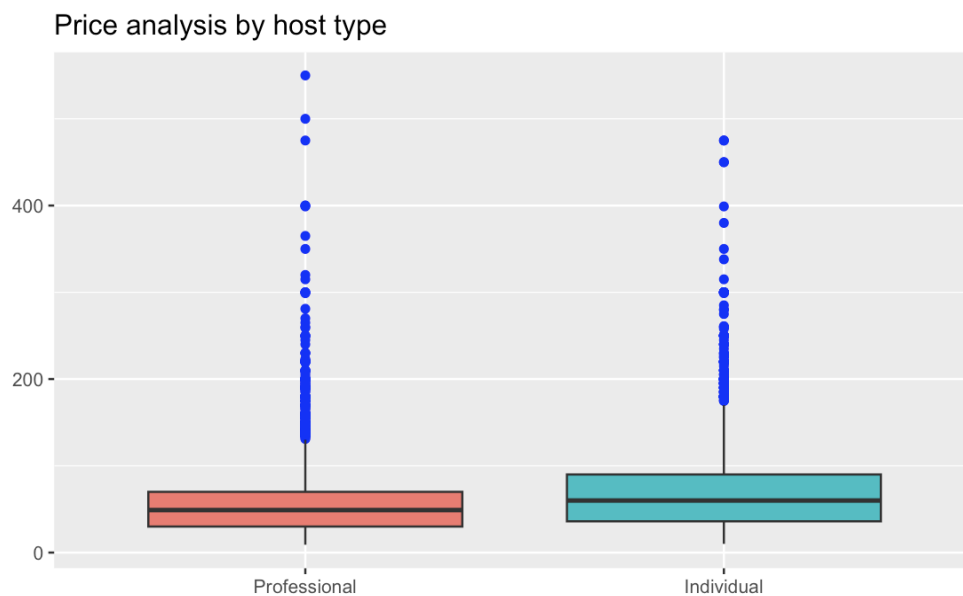


Figura 15: Análisis del precio por tipo de persona

No se ve una clara diferencia entre los precios de gestión profesional vs. Individual, por lo que se analizará si hay correlación más adelante.

- **Análisis por tipo de habitación**

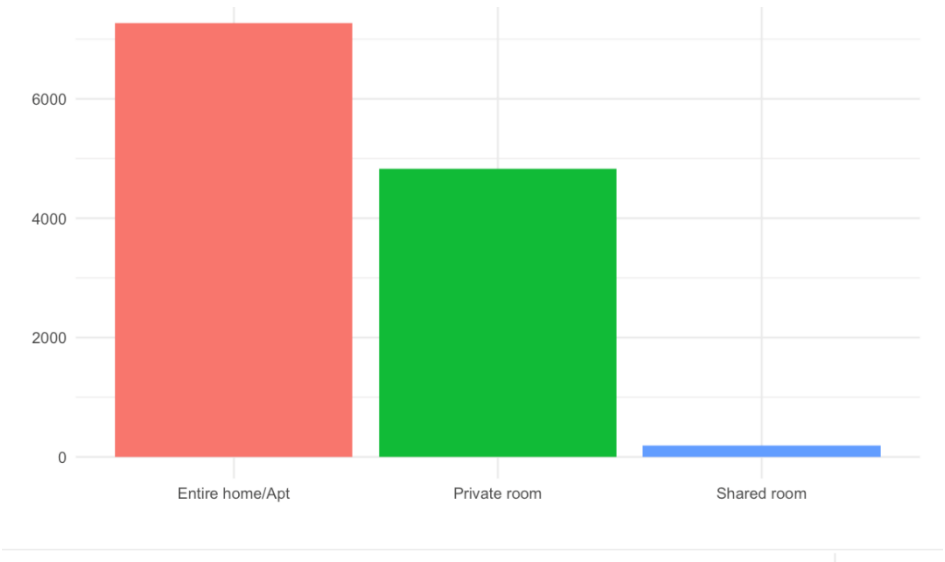


Figura 16: Análisis del precio por tipo de habitación

Se aprecia que hay más alquileres de alojamiento entero y habitación privada.

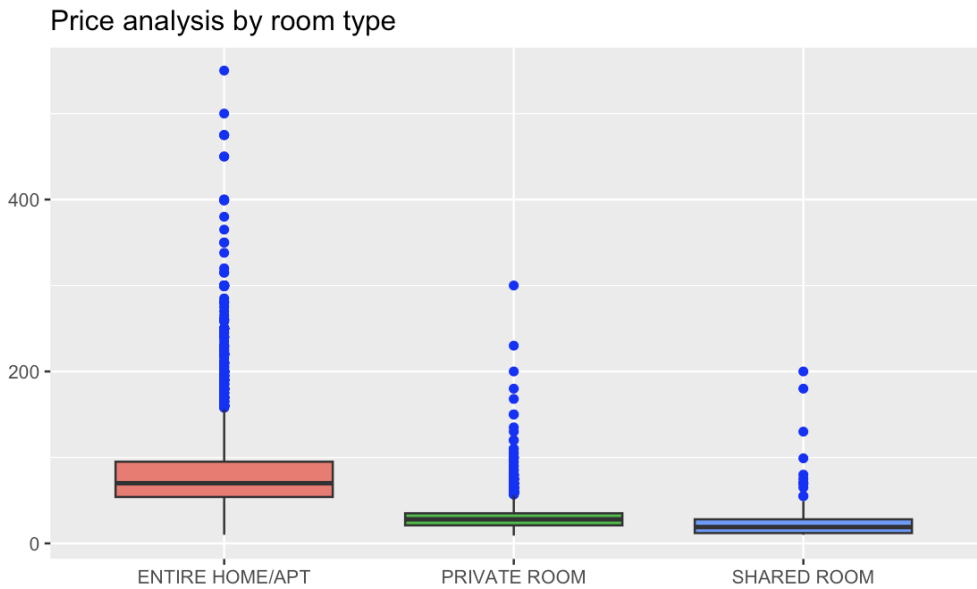


Figura 17: Análisis del precio por tipo de habitación

Se puede ver que el precio medio de un alejamiento completo está por encima de los otros tipos.

- **Análisis por tipo de propiedad**

Hay 22 tipos de propiedades, pero la mayoría se concentra en pocos tipos. Se hace una agrupación por tipo de propiedades y solo se seleccionan los tipos que tienen más de 50 viviendas. Se trabajará sólo con 6 tipos de viviendas.

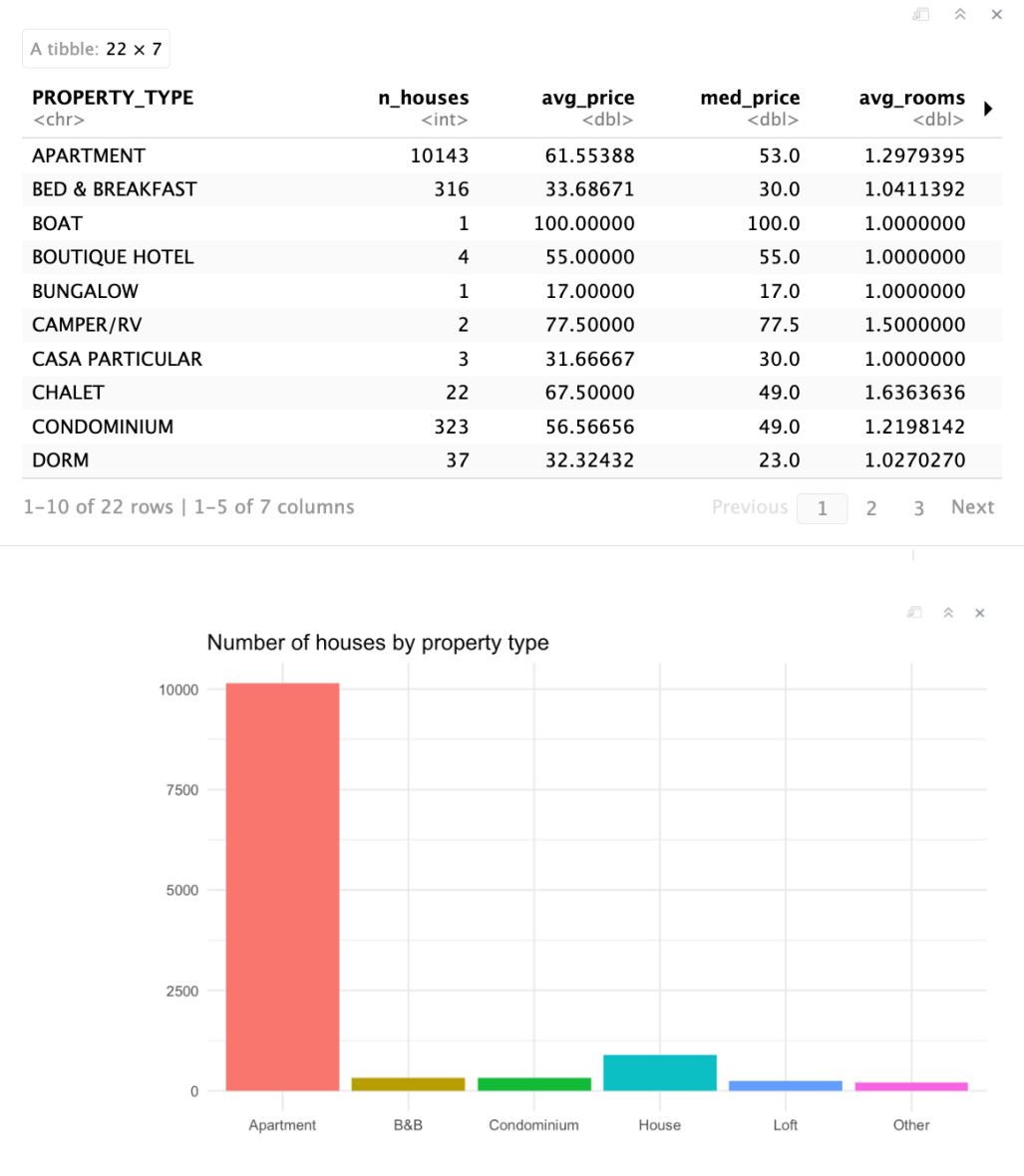


Figura 18: Análisis del precio por tipo de propiedad

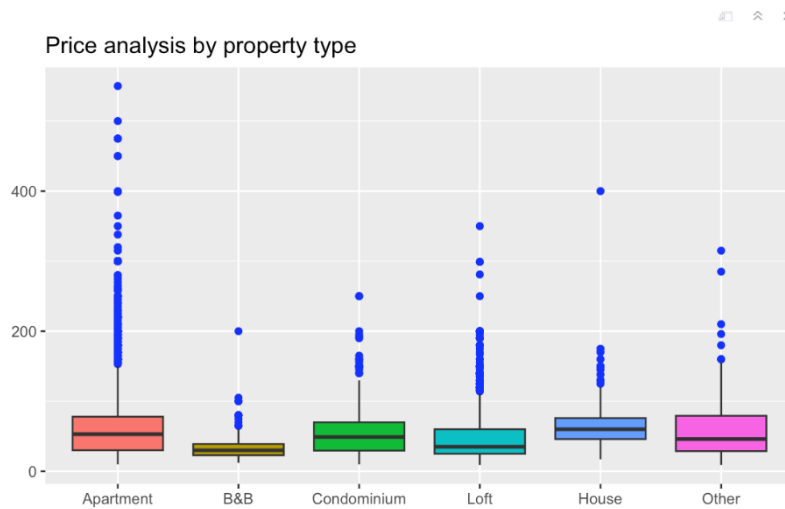


Figura 19: Análisis del precio por tipo de propiedad

Se puede ver que la mayoría de las propiedades son apartamentos y parece que el tipo de propiedad no tiene gran impacto en el precio.

- **Análisis por números de personas**

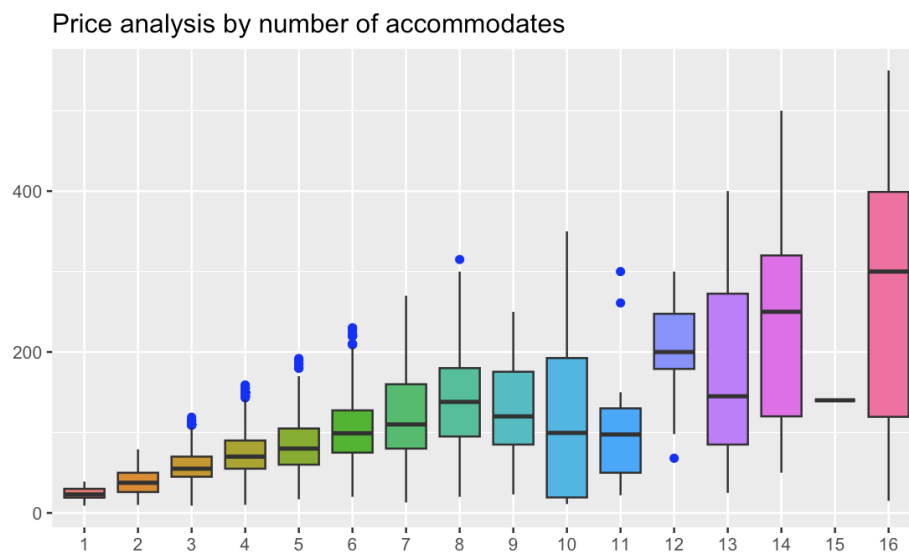


Figura 10: : Análisis del precio por número de personas

Se puede ver cómo se incrementa el precio a medida que incrementa el número de personas que acepta la vivienda.

- **Análisis por barrio**

Se realiza una agrupación por barrio en el que se encuentra la propiedad. Vemos que hay 126 barrios, pero algunos tienen pocas viviendas en alquiler.

A tibble: 126 × 7

NEIGHBOURHOOD_CLEANSED <chr>	n_houses <int>	avg_price <dbl>	med_price <dbl>	avg_rooms <dbl>
EMBAJADORES	1768	58.74774	50.0	1.2109729
UNIVERSIDAD	1253	64.17877	60.0	1.2282522
PALACIO	971	70.80330	62.0	1.3377961
SOL	885	82.98983	75.0	1.3412429
JUSTICIA	696	68.75144	60.0	1.1968391
CORTES	665	76.41654	65.0	1.2977444
TRAFALGAR	283	75.17668	60.0	1.5547703
PALOS DE MOGUER	255	51.02745	43.0	1.2666667
ARGUELLES	240	59.25000	50.0	1.4041667
PUERTA DEL ANGEL	195	36.33333	28.0	1.2666667

1-10 of 126 rows | 1-5 of 7 columns Previous 1 2 3 4 5 6 ... 13 Next

Se hace un visual de las variables agrupadas por barrio y se decide trabajar con los barrios que tienen más de 71 viviendas (tercer cuartil), trabajando con el 75% del total de los registrados.

NEIGHBOURHOOD_CLEANSED	n_houses	avg_price	med_price
Length:126	Min. : 1.00	Min. :17.00	Min. :16.00
Class :character	1st Qu.: 15.00	1st Qu.:34.62	1st Qu.:26.25
Mode :character	Median : 32.50	Median :43.97	Median :35.00
	Mean : 96.33	Mean :46.34	Mean :38.48
	3rd Qu.: 70.75	3rd Qu.:55.54	3rd Qu.:49.75
	Max. :1768.00	Max. :93.17	Max. :76.00
avg_rooms	avg_unit_rooms	avg_unit_price	
Min. :0.9286	Min. :0.3704	Min. : 8.50	
1st Qu.:1.1785	1st Qu.:0.4944	1st Qu.:14.92	
Median :1.3144	Median :0.5626	Median :17.57	
Mean :1.3182	Mean :0.5695	Mean :17.64	
3rd Qu.:1.4083	3rd Qu.:0.6262	3rd Qu.:20.56	
Max. :2.0000	Max. :0.8467	Max. :26.06	

Una vez realizado el filtro se analiza el precio de los 26 barrios.

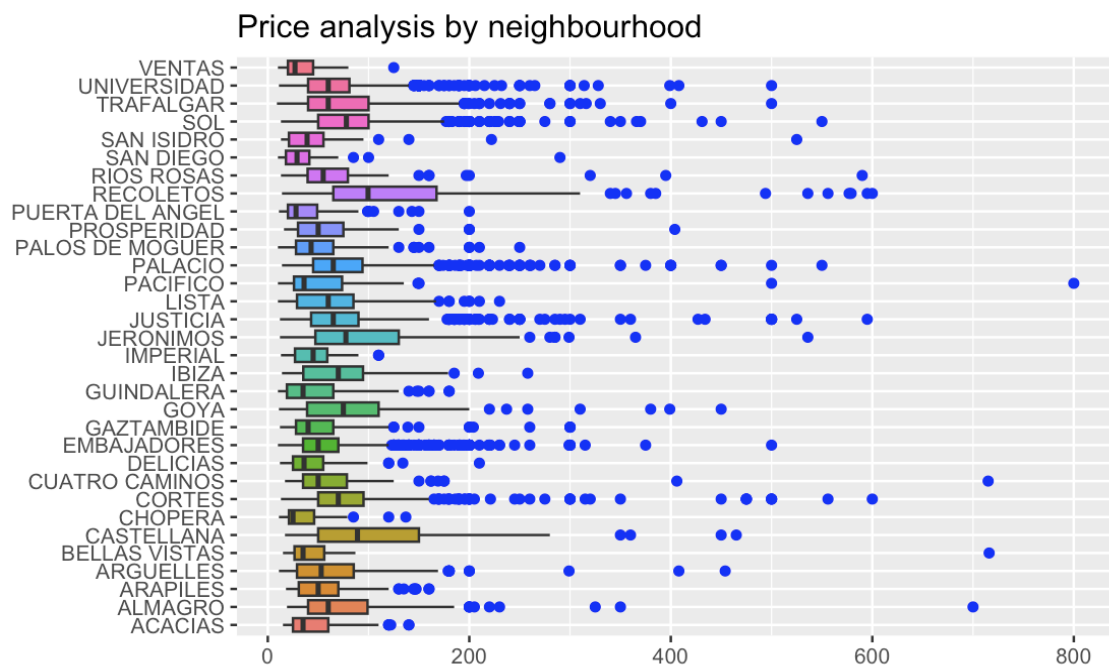


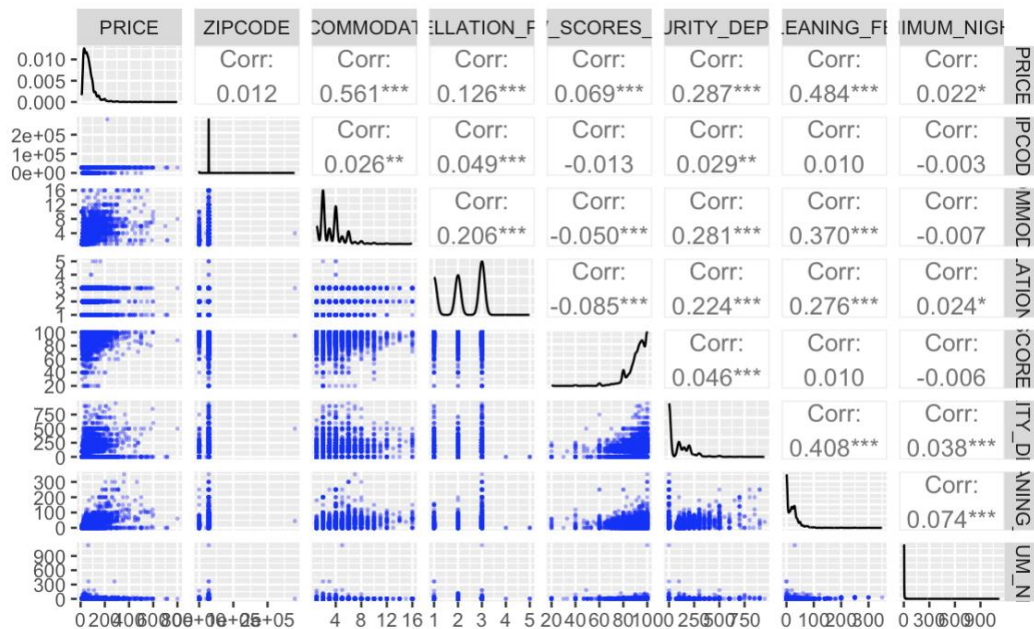
Figura 11: Análisis del precio por barrio

Se ve que hay barrios más caros como Recoletos, Jerónimos, Castellana, Goya. Con esto deducimos que el precio sí está determinado por el barrio en el que se ubica.

MODELADO PARA ESTIMACIÓN DEL PRECIO

Se va a realizar un modelo de regresión lineal para predecir el precio en función de las variables elegidas.

Se mide la correlación entre las variables que creemos que podrían influir en el precio.



Según el análisis anterior, se cree que existen múltiples variables que pueden explicar la variación en el precio y se elige para el modelo las que tienen una correlación más alta.

Call:

```
lm(formula = PRICE ~ ACCOMMODATES + REVIEW_SCORES_RATING + CANCELLATION_POLICY + SECURITY_DEPOSIT + CLEANING_FEE, data = est_price)
```

Residuals:

Min	1Q	Median	3Q	Max
-188.75	-16.31	-3.90	10.84	633.02

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-33.748962	4.412624	-7.648	2.26e-14 ***
ACCOMMODATES	12.259010	0.218302	56.156	< 2e-16 ***
REVIEW_SCORES_RATING	0.497786	0.044965	11.071	< 2e-16 ***
CANCELLATION_POLICY	-1.139844	0.516162	-2.208	0.0272 *
SECURITY_DEPOSIT	0.036771	0.003888	9.457	< 2e-16 ***
CLEANING_FEE	0.731393	0.020352	35.937	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 35.7 on 8412 degrees of freedom

(2059 observations deleted due to missingness)

Multiple R-squared: 0.5123, Adjusted R-squared: 0.5121

F-statistic: 1768 on 5 and 8412 DF, p-value: < 2.2e-16

Se ha dejado fuera del modelo el barrio en el que se encuentra, aun habiendo detectado que influye en el precio, debido a la complejidad de ese análisis, y es algo que se podrían estudiar en profundidad para mejorar el modelo.

Pese a dejar fuera el barrio, se obtiene un R – squared de 51,23%.

El modelo de regresión lineal sería el siguiente:

$$\text{Precio} = -33.749 + 12.259 \text{ Personas} + 0.4978 \text{ Rating reviews} - 1.139844 \text{ Política cancelación} + 0,0368 \text{ Fianza} + 0.7314 \text{ Gastos de limpieza}$$

- Evaluación de la calidad del modelo

Primero se separa el DataFrame en dos partes para el training y testing.

```
set.seed(1234)
idx <- sample(1:nrow(df_neigh_3Q), nrow(df_neigh_3Q)*0.7)
df_neigh_3Q.train <-df_neigh_3Q[idx,]
df_neigh_3Q.test <-df_neigh_3Q[-idx,]
```

Se calcula el modelo con el DataFrame de train.

Call:

```
lm(formula = PRICE ~ ACCOMMODATES + REVIEW_SCORES_RATING + CANCELLATION_POLICY +
    SECURITY_DEPOSIT + CLEANING_FEE, data = df_neigh_3Q.train)
```

Residuals:

Min	1Q	Median	3Q	Max
-191.14	-16.44	-3.99	10.90	634.13

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-36.202130	5.245755	-6.901	5.70e-12	***
ACCOMMODATES	12.376255	0.260894	47.438	< 2e-16	***
REVIEW_SCORES_RATING	0.524302	0.053576	9.786	< 2e-16	***
CANCELLATION_POLICY	-1.034955	0.614979	-1.683	0.0924	.
SECURITY_DEPOSIT	0.034704	0.004625	7.503	7.17e-14	***
CLEANING_FEE	0.711319	0.024052	29.574	< 2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 35.65 on 5881 degrees of freedom

(1446 observations deleted due to missingness)

Multiple R-squared: 0.5124, Adjusted R-squared: 0.5119

F-statistic: 1236 on 5 and 5881 DF, p-value: < 2.2e-16

Ahora se calculan las figuras de calidad, tanto en training como en testing.

```
df_neigh_3Q.train$price_est <- predict(model_df_neigh_3Q, df_neigh_3Q.train)
caret::postResample(pred = df_neigh_3Q.train$price_est, obs=df_neigh_3Q.train$PRICE)
```

RMSE	Rsquared	MAE
35.6281288	0.5123577	21.0923318


```
df_neigh_3Q.test$price_est <- predict(model_df_neigh_3Q, df_neigh_3Q.test)
caret::postResample(pred = df_neigh_3Q.test$price_est, obs=df_neigh_3Q.test$PRICE)
```

RMSE	Rsquared	MAE
35.8463648	0.5119811	21.1863208


```


```

Se aprecia que el R – squared es parecido en el testing y training, en torno al 0,51. Se considera que el modelo es aceptable y, para asegurar, se procede a analizar los residuos.

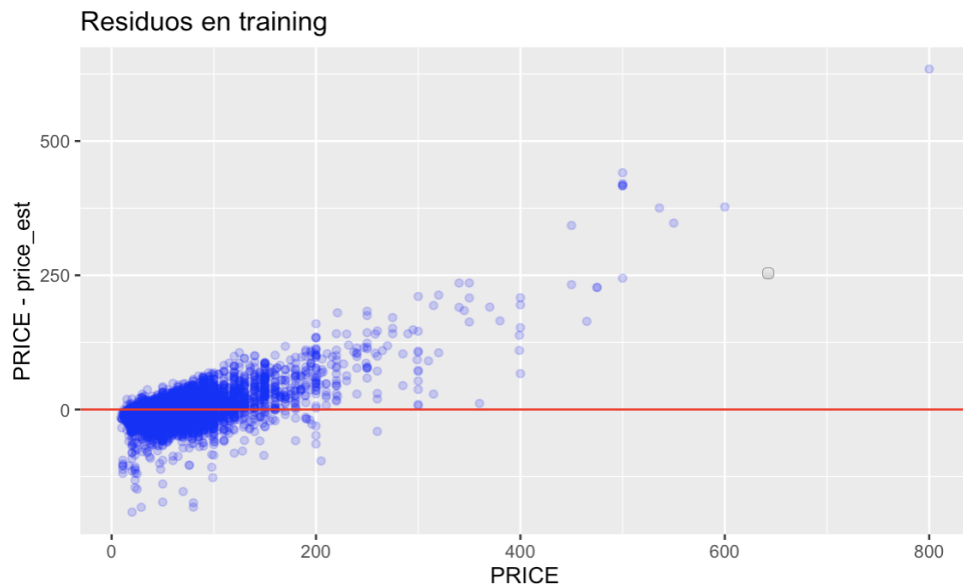


Figura 12: Residuos en training

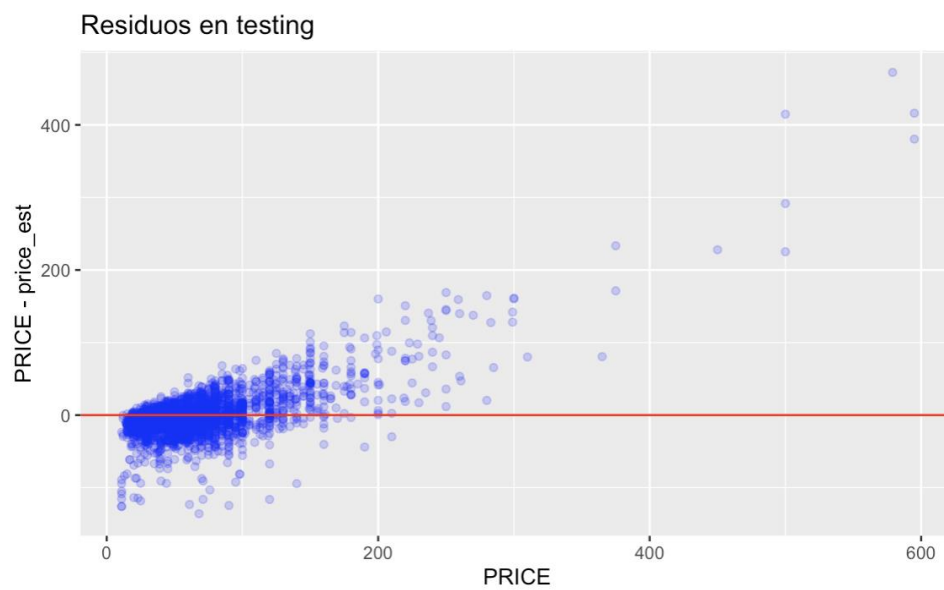


Figura 13: Residuos en testing

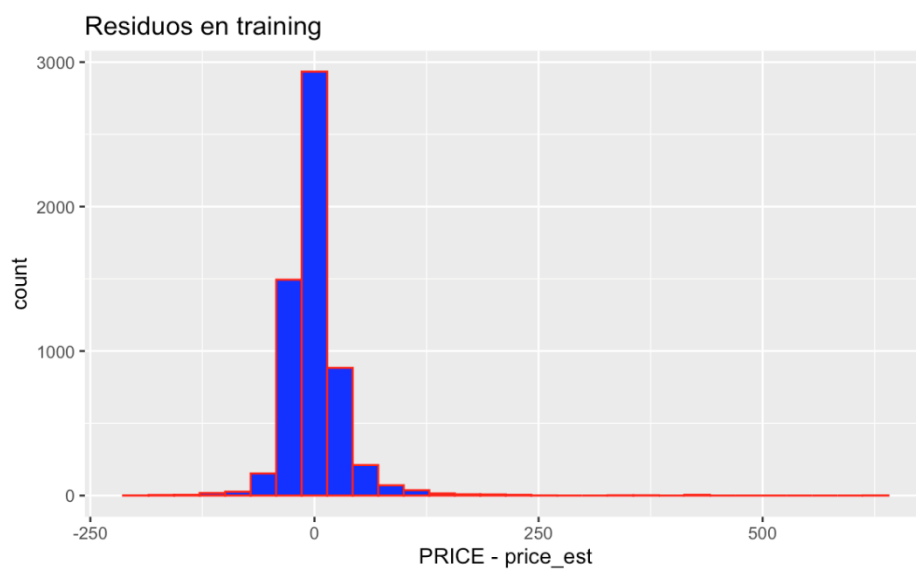


Figura 14: Residuos en training

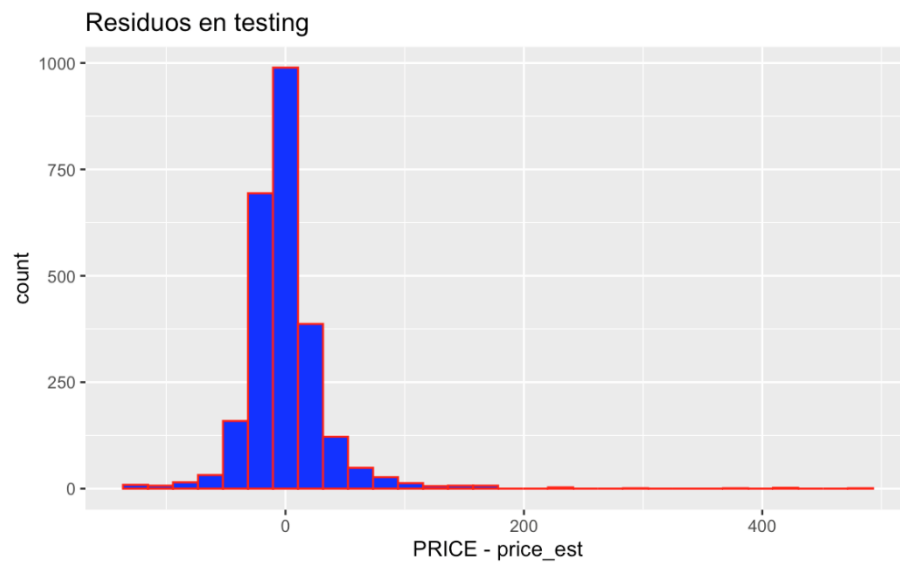


Figura 15:Residuos en testing

Los residuos siguen una distribución normal y los valores medios de los residuos están centrados en 0, por lo que se considera que el modelo es correcto.

CONCLUSIONES

- **Suposiciones iniciales: ¿Cuáles han demostrado ser válidas y cuáles no? ¿Por qué?**

Detectamos en el DataFrame un ID de hospedador y al agruparlo, vimos que hay hospedadores que gestionan varias viviendas. Debido a que hay empresas que se dedican a gestionar viviendas de alquiler vacacional, asumimos que los anuncios de esas viviendas estaban gestionados por profesionales. Nos interesaba conocer si en este caso el precio medio se incrementaba respecto al precio de viviendas de particulares.

Después del análisis vimos que la mayor parte de viviendas era de gestión profesional, sin embargo, no influía en el precio del alquiler.

Las principales variables que vimos que influían en el precio son el número de personas que aceptaba la vivienda, la política de cancelación, la fianza y si se cobraba aparte una tasa de limpieza.

- **Métricas seleccionadas: ¿Han sido las correctas o no? ¿Por qué?**

Gracias al exhaustivo estudio que se realizó en los primeros bloques del proyecto, se pudieron identificar las variables más precisas y relevantes para el estudio. Se primaron aquellas variables con menos datos NaN o aquellas que aún con datos NaN, se pudieran inferir valores sustitutivos, además de aquellas que aportasen información relevante para el objetivo del proyecto.

Sin embargo, durante el proceso, se evaluaron cambios de métricas para un estudio más preciso:

- Tasa de alquiler / Ratio de ocupación

Al comenzar el ejercicio la variable de Tasa de Alquiler / Ratio de Ocupación / Ratio de Alquiler fue detectada y elegida como variable principal del estudio. Sin embargo, su cálculo fue inicialmente considerado de la siguiente manera:

$\text{Ratio_alquiler} = \text{monthly_reviews} / (30 / \text{minimum nights})$

Una vez comenzado el análisis, se observó que dicho cálculo no era correcto ya que la variable con la que se contaba del dataset inicial se trataba de una media mensual y, además, los minimum nights parecían no haberse mantenido estables durante el periodo en el que la vivienda estaba disponible en Airbnb puesto que había viviendas con más

reviews que días había estado disponible según el periodo mínimo de alquiler que notificaba en la aplicación.

Por lo tanto, una nueva ratio de ocupación más preciso tuvo que ser definida, tal y como se detalla en el apartado de “Análisis Exploratorio de Datos” que tuviera en cuenta los días reales que el alojamiento ha estado disponible, así como las reviews aportadas, excluyendo aquellos datos cuyos resultados fueran incompatibles con lo especificado en el portal de Airbnb.

- **Teniendo en cuenta lo aprendido: ¿Qué cosas se harían igual y cuáles se harían de otra forma? ¿Por qué?**

Empezamos el análisis y exploración del dataframe para la limpieza en R, pero decidimos cambiar a Python, debido a que las librerías de Pandas, Janitor, Unidecode, nos simplificaron mucho el proceso de limpieza. En R la limpieza estaba siendo muy costosa, por lo que, si tuviéramos que realizar otra vez este proyecto, empezaríamos con Python.

Otro de los puntos a mejorar, sería empezar el trabajo con una idea clara de lo que queremos investigar. Empezamos el análisis sin una idea muy clara, explorando, añadiendo y calculando variables que a posteriori no se utilizaron. Creemos que partiendo de una idea bien definida el trabajo es mucho más eficiente.

- **Conclusiones y lecciones aprendidas.**

Las viviendas con ratios de alquiler más altos, son menos susceptibles a percibir un cambio en la ratio de alquiler al modificar otras variables que las viviendas con ratios de alquiler menos altos.

Por ejemplo, las viviendas con ratios de alquiler más bajos, son más propensas a percibir un cambio positivo en su ratio de alquiler ante reviews más positivas y sufren cambios negativos más exponenciales en su ratio de alquiler cuando se incrementa el precio medio por persona.

Esto puede explicarse por otros factores que pueden justificar el alquiler de la vivienda, ya que, si una vivienda es más atractiva y por lo tanto tiene una ratio de alquiler medio alto, percibirá un menor impacto ante el cambio de cualquier otra variable, mientras que,

si una vivienda cuenta con una demanda baja de por sí, cualquier variable que afecte positiva o negativamente dicha demanda, verá un mayor efecto en su ratio de alquiler, ya que su demanda es mucho más elástica.

Por lo tanto, aquellos propietarios que cuenten con una vivienda clasificada como ratio de alquiler bajo, podrán obtener del dashboard una estrategia más directa y con acciones más inmediatas que aquellos que ya cuentan con una demanda alta.

Respecto al modelo de regresión lineal, pese a obtener un modelo que predice el precio hasta en un 51% de las veces, no se ha utilizado la variable precio en el modelo debido a su complejidad y a la necesidad de un análisis mucho más detallado, y después de haber deducido por otros análisis que sí incluye en la determinación del precio, consideramos que es una mejora que se puede hacer a la precisión del modelo.

BIBLIOGRAFÍA

1. (<https://public.opendatasoft.com/explore/dataset/airbnb-listings>)
2. <https://docs.google.com/spreadsheets/d/1iWCNJcSutYqpULSQHINyGInUvHg2BoUGoNRIGa6Szc4/edit#gid=1322284596>