PROYECTO FINAL BOOTCAMP MUJERES EN TECH

Especialidad DATA

2022-2023

GRUPO 15:

Nerea Castañares
Olga Marín
Sandra Moreno
Jessica Piñas
Carmen Santaló

1. Definición del Objetivo y alcance el proyecto

El objetivo de este proyecto es analizar los precios de los alojamientos de Airbnb en Madrid y determinar las variables que influyen en su valoración.

El proyecto se enfocará en analizar los precios de los alojamientos de Airbnb en Madrid durante entre los años 2012 y 2017. Se utilizarán técnicas de análisis de datos y minería de datos para identificar las variables más relevantes que influyen en el precio de los alojamientos, como la ubicación, las opiniones de los inquilinos, el tipo de alojamiento y algunas características.

Se espera que el proyecto genere un modelo predictivo que pueda ser utilizado por futuros propietarios de alojamientos de Airbnb en Madrid para establecer precios más precisos y competitivos.

a. Suposiciones Iniciales

Se ha partido de las siguientes hipótesis para la realización de este análisis:

- La ubicación del alojamiento y el número de habitación y camas son factores determinantes en el precio de los alojamientos.
- El precio del alojamiento es superior en los barrios con mayor renta per cápita.
- El número de reviews positivas influye en el precio del alojamiento.

b. Métricas seleccionadas

El principal KPI para la realización de este proyecto ha sido el **precio** de los alojamientos.

Por otro lado, hemos tenido en cuenta otras métricas para valorar el éxito del análisis, como por ejemplo:

- La correlación entre diferentes variables: Con este análisis hemos podido determinar qué variables tienen más influencia en el precio de los alojamientos.
- Análisis del coeficiente de determinación R2: A través del coeficiente R2 evaluaremos la capacidad del modelo para determinar la variación de los precios de alquiler de los alojamientos de Airbnb en Madrid.

2. Metodología

Para realizar el análisis de precios de los alojamientos de Airbnb se ha seguido la siguiente metodología:

1. **Recopilación de datos.** La fuente de datos principal es un excel con información sobre diferentes propiedades con anuncio en airbnb. Además hacemos uso de datos oficiales del INE para enriquecer nuestro análisis.
2. **Extracción, transformación y carga de datos (ETL)**. Los datos en bruto sufren un proceso de transformación y limpieza para poder ser analizados. Además, hacemos uso de una base de datos para cargarlos. Para este proceso hacemos uso de Python, principalmente de las librerías Pandas y Numpy.
3. **Análisis exploratorio de los datos**. Miramos y analizamos mediante gráficos y tablas los datos, para poder extraer información y posibles relaciones entre variables. Este apartado se ha desarrollado en Python con ayuda de la librería gráfica Matplotlib.
4. **Visualización de datos**. Con Tableau creamos dashboards que recopilan de forma interactiva y visual la relación entre nuestro KPI y algunas variables de interés.
5. **Modelo de regresión lineal**. Todo este proceso culmina en la creación de un modelo de regresión lineal en R, donde se intenta predecir el precio de un inmueble en airbnb según ciertas variables.

1. Recopilación de datos

La fuente de datos principal es un Excel sacado de opendatasoft. Esta base de datos contiene información sobre los alojamientos con anuncios de airbnb. Nosotras recibimos un dataset de 14.780 filas con 89 columnas, ya que se le ha aplicado un filtro en el que solo se guardan aquellas que contienen la palabra “Madrid”.

Podemos ver que el archivo tiene muchas columnas que en su mayoría no vamos a usar, por lo que tomamos la decisión de reducir su número. En su mayoría se han descartado columnas conformadas por textos largos y muy variables, como descripciones. Esto se debe a la dificultad que presenta el poder estandarizar estas respuestas e incluso al poco valor que podrían tener, por su alta subjetividad. Tras el análisis del CSV proporcionado, nos hemos quedado con 39 de las 89 columnas, que pasarán a ser limpiadas para su posible uso en pasos posteriores.

Por otra parte, para enriquecer nuestro análisis hemos cogido datos del INE sobre la renta media por barrio en 2017.

2. ETL

En esta fase se ha realizado la subida de los datos seleccionando las columnas del excel elegidas en la fase de recopilación de datos, se ha procedido a la limpieza de estos datos y su posterior creación de un script de subida a una base de datos PostgreSQL y descarga de los datos en CSV. Todo el proceso se ha realizado en Python con las librerías Pandas y Numpy y la creación y carga a la base de datos con scripts SQL.

Lo primero ha sido la carga de los datos, que se han pasado del Excel a un Dataframe en pandas. Sin embargo, como solo son necesarias 39 columnas, descartamos el resto para hacer el dataframe más ligero y legible. Además, en los datos originales se han colado observaciones que no son de la Comunidad de Madrid, si no que en alguna de sus columnas tienen la string “Madrid” en algún lado. Por lo tanto, hemos filtrado los datos según la columna “State”, que en el caso de España se refiere a la comunidad autónoma. Así pues, nos quedamos únicamente

con las observaciones que tengan la string “Madrid” o “madrid” (por “Comunidad de Madrid” o “comunidad de madrid”).

Pese a que la carga se ha realizado correctamente, hemos observado se ha inferido el tipo de dato para cada una de las columnas y en algunos casos se ha hecho erróneamente. Nos encontramos con que muchas de las variables de número entero son ahora decimales, y las fechas se encuentran como texto plano. Así, para poder trabajar mejor y limpiar estos errores de tipología del dato, hemos corregido estas diferencias.

Una vez realizados estos cambios, podemos proceder a limpiar algunas variables, principalmente numéricas.

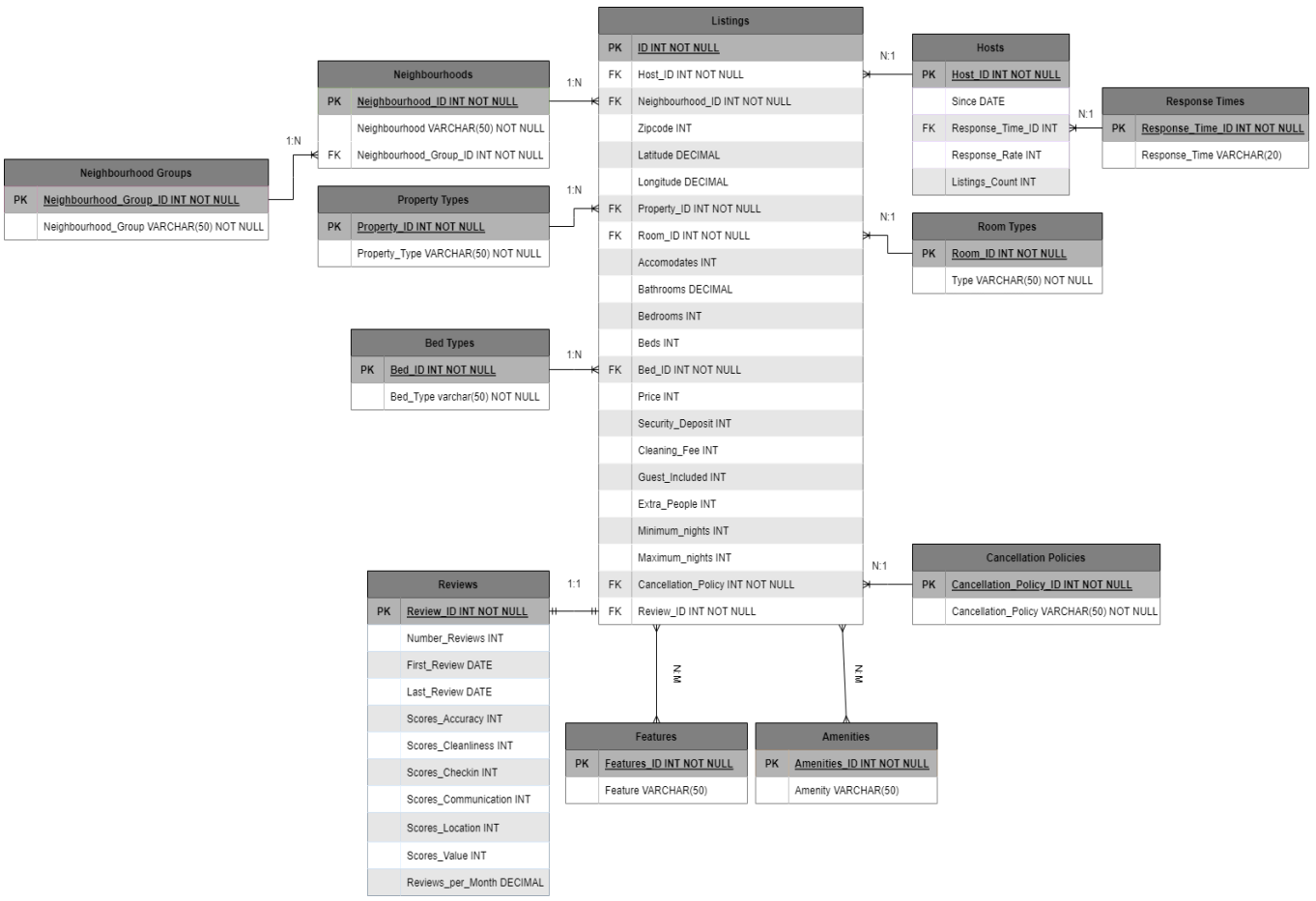
Lo primero es que hemos detectado que algunos Códigos postales son erróneos, ya que en España tienen 5 cifras, y algunos no cumplen esa restricción. Así, hemos convertido estos códigos a nulo para que no creen interferencias.

Lo siguiente ha sido un análisis de nulos, ya que en muchos casos en las variables numéricas, esos nulos pueden ser convertidos a 0. Podemos ver que las variables de ‘Security Deposit’ tiene un 57,50% de nulos, seguramente porque no hay. Algo parecido pasa en ‘Cleaning Fee’, con un 40,88% de nulos. Como en ambas estamos hablando de cantidades de dinero, estos nulos equivalen a decir que estas tasas son 0, por lo que hacemos el cambio.

Por otro lado, las variables relacionadas con ‘Reviews’ presentan alrededor de un 20% de nulos cada una. En estos casos, no lo vamos a cambiar por 0 ya que estos nulos se corresponden a la falta de review, y no que se les haya puesto un 0.

Finalmente observamos que en nuestra variable principal, el precio, hay 9 observaciones nulas. Vamos a descartarlas todas, ya que no tiene sentido analizarlas y podrían emborronar nuestras conclusiones.
Habiendo limpiado los datos y la presencia de nulos, es el momento de empezar a maquetar nuestra base de datos. La idea es dividir el dataframe que tenemos en varios normalizados, para luego poder insertarlos como tablas en nuestra base de

datos. Para esto, necesitamos primero tener claro cuál va a ser nuestro modelo normalizado. En la siguiente imagen tenemos el modelo, con las relaciones entre tablas indicadas.



Podemos apreciar como es necesario separar el dataframe original en varios, y además normalizar lo que será en un futuro la tabla listings con todos los ID del resto de tablas necesarias. Ese es nuestro primer paso, donde creamos un data frame por cada tabla que se muestra en el diagrama, ya con las columnas apropiadas seleccionadas. Sin embargo, estos no están normalizados, ya que por

ejemplo el listings hace falta pasar de las variables categóricas que están actualmente a los ID de las que contienen esos datos. Otro problema lo plantean las tablas features y amenities, ya que en los datos originales, esas columnas aparecen como una cadena de texto de elementos separados por comas (por ejemplo, una observación de la columna amenities es: “TV, Wireless Internet, Kitchen, Breakfast, Heating, First aid kit, Essentials, Shampoo, Hangers”) y nosotras queremos una tabla que refleje todas las posibilidades de entre esta lista, con un elemento por fila. Además, necesitamos crear dos tablas relacionales N:N entre listings-amenities y listings-features.

El Jupyter Notebook ‘ETL’ recoge todo este proceso, y las tres funciones principales solventan cada uno de los problemas.

```python
def separate_commas(dataframe, column):
    """Recoge en una lista todas las categorias de una columna de un dataframe separada por comas"""
    values = list()
    raw = dataframe[column].unique().tolist()
    for item in raw:
        if type(item) == str:
            separated = item.split(", ")
            for i in separated:
                if i not in values:
                    values.append(i)
    return values

def string_to_index(dataframe, col, reference, index):
    """Cambia los valores de una columna por los indices correspondientes de otra"""
    for value in dataframe[col].unique().tolist():
        i = reference.loc[reference[col] == value, index].values[0]
        dataframe.loc[dataframe[col] == value, col] = i
    return dataframe[col]

def explode_dataframe(relational, reference, column):
    """Crea un dataframe al estilo de una tabla relacional N:N"""
    relational[column] = relational[column].apply(lambda x: x.split(', ') if type(x) == str else np.nan)
    relational = relational.explode(column).reset_index()
    j = 0
    for i, row in relational.iterrows():
        index = reference.loc[reference[column] == row[column], "index"]
        relational.at[i, 'index'] = j
        j += 1
        if not index.empty:
            relational.at[i, column] = index.values[0]
    return relational
```

La función separate_commas(dataframe, column) crea una lista a partir de una columna donde cada observación es una string separada por comas. En esta lista aparece una vez cada posible valor, por lo que luego podemos crear un dataframe a partir del retorno de la función.

La función string_to_index(dataframe, col, reference, index) convierte las valores de una columna en los índices que tienen en otro dataframe, lo que nos permite normalizar las Foreign keys de una tabla con las Primary Keys de otra.

Finalmente, explode_dataframe(relational, reference, column) hace uso del método .explode() de pandas para crear un dataframe al estilo tabla relacional N:N.

Por ejemplo la siguiente observación de un posible dataframe:

ID	amenity
1	TV, Wireless Internet, Kitchen

Pasaría a ser:

ID	amenity
1	TV
1	Wireless Internet
1	Kitchen

Así pues, tras modificar los dataframes y hacer estos cambios, es el momento de usarlos. Para poder hacer uso de estas tablas en pasos posteriores, nos descargamos cada uno de los dataframes en formato CSV, además del dataframe original ya limpio y con las columnas necesarias únicamente.

Para usar toda la infraestructura que hemos creado para insertar los datos en PostgreSQL, el último paso es automatizar la creación de un script SQL de inserción de datos. Así, el Jupyter Notebook acaba creando un archivo “data_upload.sql” en el

que inserta línea por línea los “INSERT INTO() VALUES()” necesarios por cada dataframe. Previamente, hemos creado un script SQL que crea las tablas siguiendo el esquema presentado anteriormente.

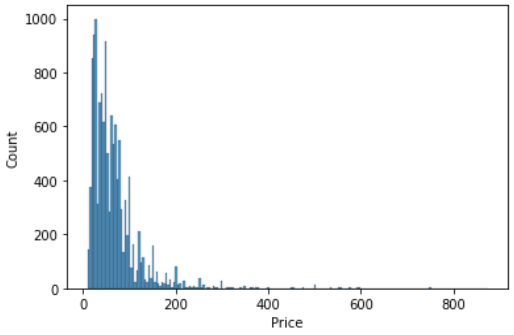
Nuestra ETL queda por lo tanto finalizada, con los datos limpios cargados en PostgreSQL y descargados como CSV para poder usarlos en el análisis.

3. Análisis Exploratorio de datos

Con los datos ya limpios, vamos a realizar el análisis exploratorio de datos utilizando Python, con las librerías Pandas, Matplotlib, Seaborn y Scipy.

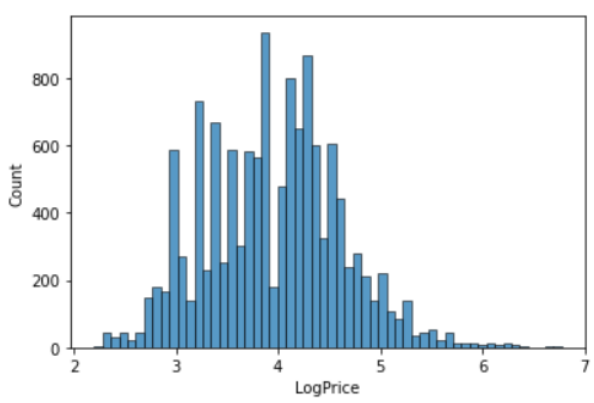
Cargamos el CSV de los listings ya limpio en el paso anterior a un dataframe de Pandas.

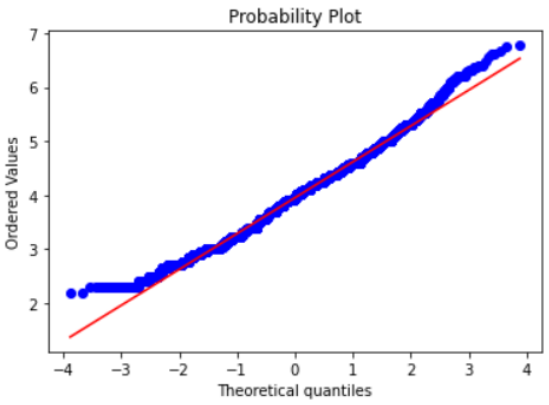
Primero hacemos un análisis de la variable precio que es nuestro KPI principal, con un histograma y una gráfica de probabilidad normal.



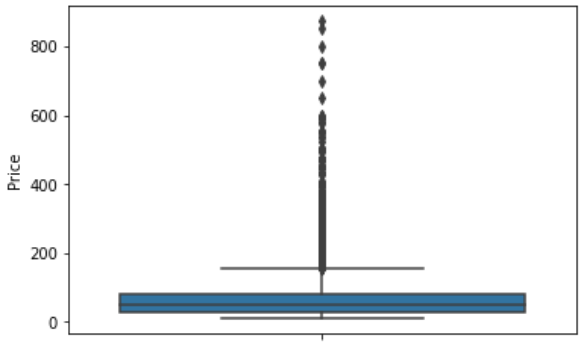
Vemos que la variable precio no sigue una distribución normal, muestra una asimetría positiva, picos y no sigue la diagonal de la gráfica de probabilidad.

Creamos una nueva columna con los precios en escala logarítmica, y haciendo otro histograma vemos que sí sigue una distribución normal, posteriormente a la hora de hacer el modelo predictivo será interesante ver si usando el precio en logaritmo mejora el modelo.



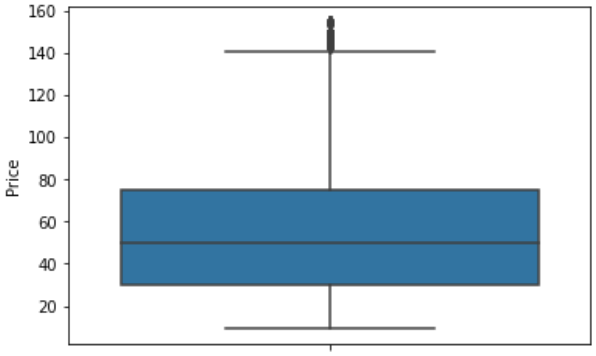


El siguiente paso es hacer un boxplot de la variable Precio. Rápidamente vemos la presencia de outliers.



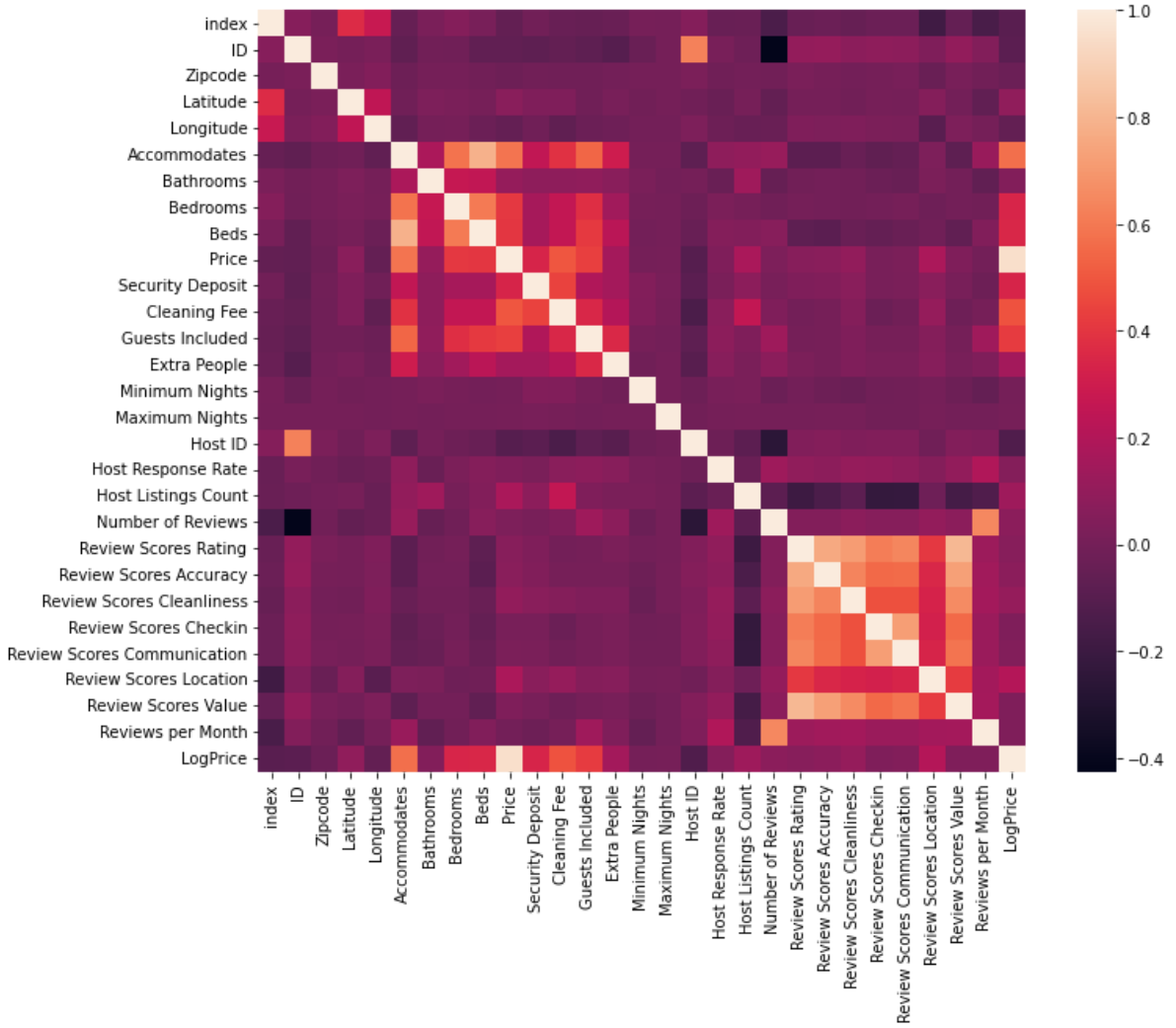
Para eliminar los outliers vamos a calcular el primer cuartil (Q1) y el tercer cuartil(Q3) y posteriormente el rango intercuartílico (IQR) restando Q1 a Q3.

Eliminamos los datos que se queden fuera de 1.5 veces el IQR.



Ahora vamos a pasar a hacer un análisis multivariable, para ello es interesante hacer una matriz de correlación de nuestros datos calculando el coeficiente de correlación entre cada par de variables.

Para visualizarlo mejor vamos a crear un mapa de calor a partir de estas correlaciones. Estas correlaciones aparecen en el mapa de color con distintos tonos, con los colores más claros representando una mayor correlación.

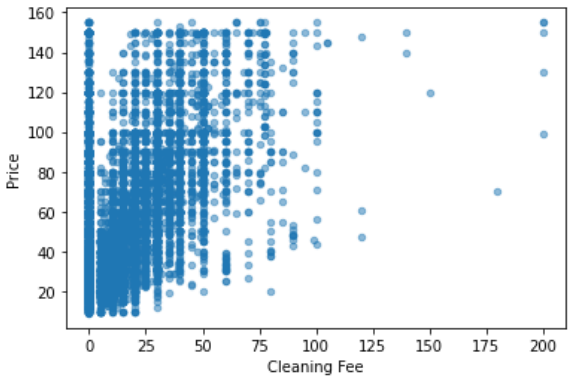


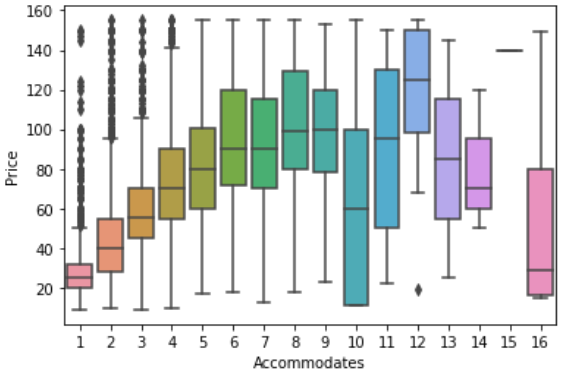
A simple vista observamos que las variables que parecen tener una mayor correlación con la variable precio son Accommodates, Bedrooms y Cleaning fee.

La variable Accommodates se refiere al número de personas que la vivienda puede acomodar, esta variable vemos que está intensamente correlacionada con la variable Beds (el número de camas en la vivienda), una variable que en un principio suponíamos que afectaría más al precio.

La variable Cleaning Fee hace referencia a la tarifa de limpieza y la variable Bedrooms al número de habitaciones en la vivienda. El gráfico de correlación indica que es una de las variables que más afectan, si nos fijamos en el siguiente gráfico

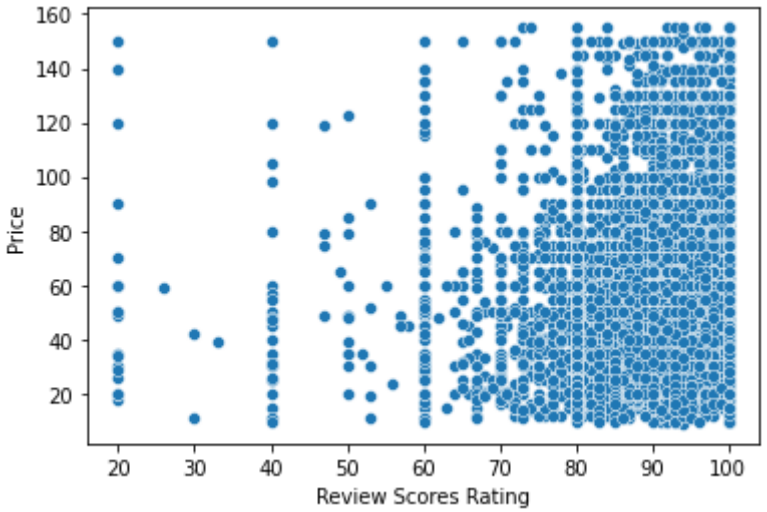
se observa que a medida que aumenta el precio del alquiler también lo hace el de la tasa de limpieza



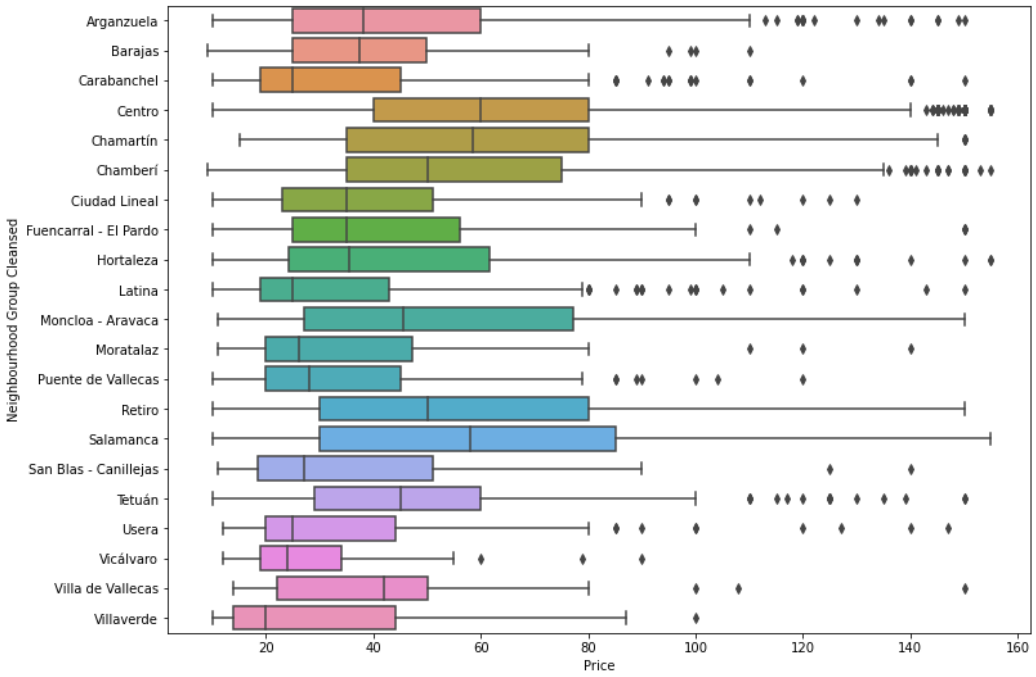


Una de nuestras suposiciones iniciales era que un número elevado de reviews positivas influiría en un precio mayor, pero tras analizar la variable Review Scores Rating y el precio en un boxplot vemos que no es necesariamente cierto.

También hacemos un gráfico de dispersión entre el precio y el número de reviews, y lo que sí apreciamos es que a mayor precio también parece haber menor número de reviews, lo que puede significar que las viviendas más caras se alquilan menos.



La ubicación del alojamiento es otra de las variables que influye en el precio, como se observa en la siguiente gráfica:



Otra de nuestras hipótesis era que la ubicación también influía en el precio y que los barrios con mayor renta per cápita eran los barrios con los alojamientos más caros.

Para comprobar esto nos descargamos a través de la página del INE un csv que contiene la renta media de los hogares de los distintos barrios de Madrid en el año 2017.

Comparamos los barrios con mayor y menor renta media con los barrios con mayor y menor precio medio de nuestro dataset, y podemos comprobar que sí parece haber relación, algunos de los barrios con rentas más altas también son los barrios con precios más caros.

```
: # Precios medios de los distintos tipos de Neighbourhood

N_Prices = data_clean.groupby(['Neighbourhood Cleansed']).Price.mean()
N_Prices = N_Prices.reset_index()
N_Prices=N_Prices.sort_values('Price',ascending=[0])
print(f'\nPrecios más Altos:\n\n{N_Prices.head(10)}\n')
print(f'Precios más Bajos:\n\n{N_Prices.tail(10)}')
```

```
Precios más Altos:

    Neighbourhood Cleansed      Price
98                Recoletos  81.193548
47              El Plantío  78.333333
29              Castellana  75.493151
112                    Sol  74.538550
59          Hispanoamérica  73.173913
62               Jerónimos  72.986842
39                  Cortes  70.349003
75           Nueva España  68.980769
60                  Ibiza  68.891089
56                   Goya  68.004926

Precios más Bajos:

    Neighbourhood Cleansed      Price
```

```
rentas = pd.read_csv('30677bsc.csv', sep=(';'), encoding='latin-1', decimal= ('.'))
rentas['Total'] = rentas['Total'].apply(lambda x: x.replace('.',''))
rentas['Total'] = rentas['Total'].apply(lambda x: x.replace(',','.')).astype('float').astype("Float64")
rentas = rentas.drop(['Sexo'], axis=1)
rentas = rentas.sort_values(['Total'], ascending=False)
print(f"\nRentas más Altas: \n\n{rentas['Nivel territorial: Nivel 2'].head(10)}\n")
print(f"Rentas más Bajas: \n\n{rentas['Nivel territorial: Nivel 2'].tail(10)}")
```

```
Rentas más Altas:

24                        El Viso (Madrid)
106                       Piovera (Madrid)
18                     Recoletos (Madrid)
23                    Castellana (Madrid)
28                  Nueva España (Madrid)
55     Aravaca-Plantio-Valdemarin (Madrid)
114                       Palomas (Madrid)
50                    Mirasierra (Madrid)
17           Niño Jesús-Jerónimos (Madrid)
39                        Almagro (Madrid)
Name: Nivel territorial: Nivel 2, dtype: object

Rentas más Bajas:

115     San Andrés-2 (Madrid)
129           Hellín (Madrid)
127           Ambroz (Madrid)
90         Numancia-1 (Madrid)
81         Pradolongo (Madrid)
82          Entrevías (Madrid)
83        San Diego-1 (Madrid)
135           Amposta (Madrid)
140       San Diego-2 (Madrid)
117    San Cristobal (Madrid)
Name: Nivel territorial: Nivel 2, dtype: object
```

4. Visualización de datos

Utilizando Tableau, vamos a crear los dashboards para la visualización de los datos.

Primero decidimos qué KPIs concretos queremos mostrar y creamos los campos calculados y parámetros necesarios para poder profundizar en puntos de datos específicos.

4.1. Dashboard de Precio

Empezamos con el dashboard del precio:

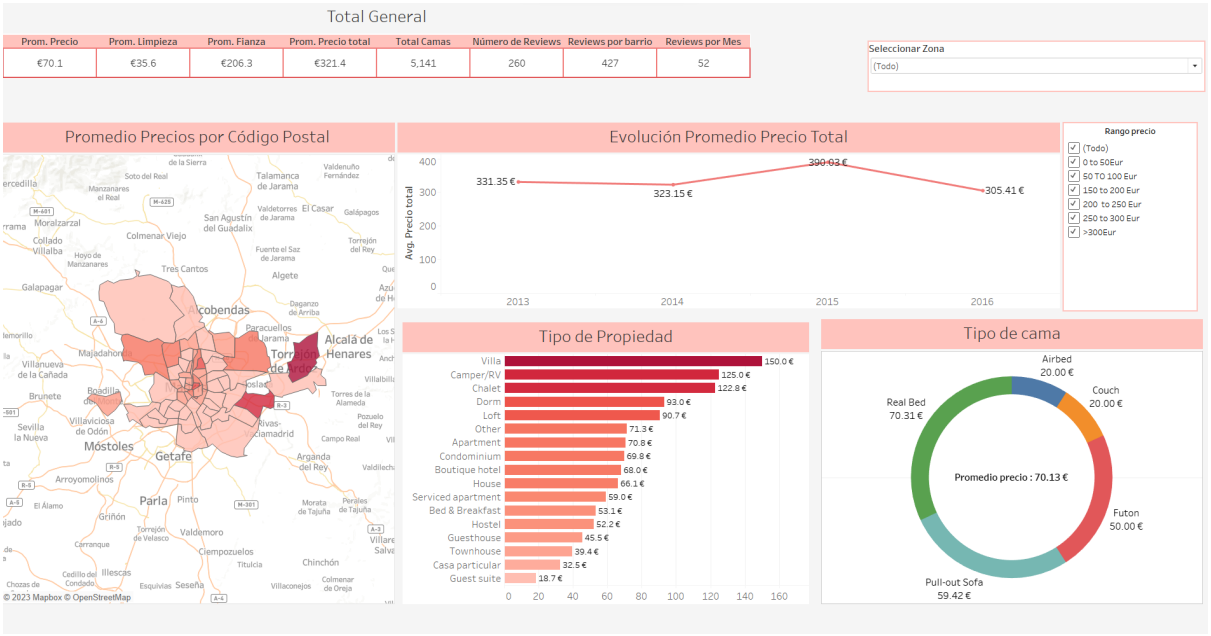
En la parte superior del dashboard hacemos un resumen de las variables más relevantes: Precio, Reviews y número de camas.

Añadimos un mapa con el promedio de precio por código postal y la información del precio a detalle de zona y barrio. El mapa también se puede usar para filtrar la información de las vistas de todo el dashboard.

Añadimos un gráfico con la evolución promedio del precio total, después añadimos un gráfico de barras que muestre el promedio de precio por tipo de propiedad organizado de mayor a menor, que también se podrá utilizar para filtrar la

información de todas las vistas del dashboard y finalmente un gráfico de anillo que muestre el promedio de precio por tipo de cama.

Ahora decidimos qué filtros vamos a necesitar, aplicamos un filtro por zona y creamos un campo calculado que utilizaremos como filtro por rango de precio, ambos filtros están relacionados y aplican a todas las vistas del dashboard, de esta forma podremos recopilar la información por zona y por rango de precio.



4.2. Dashboard de Reviews

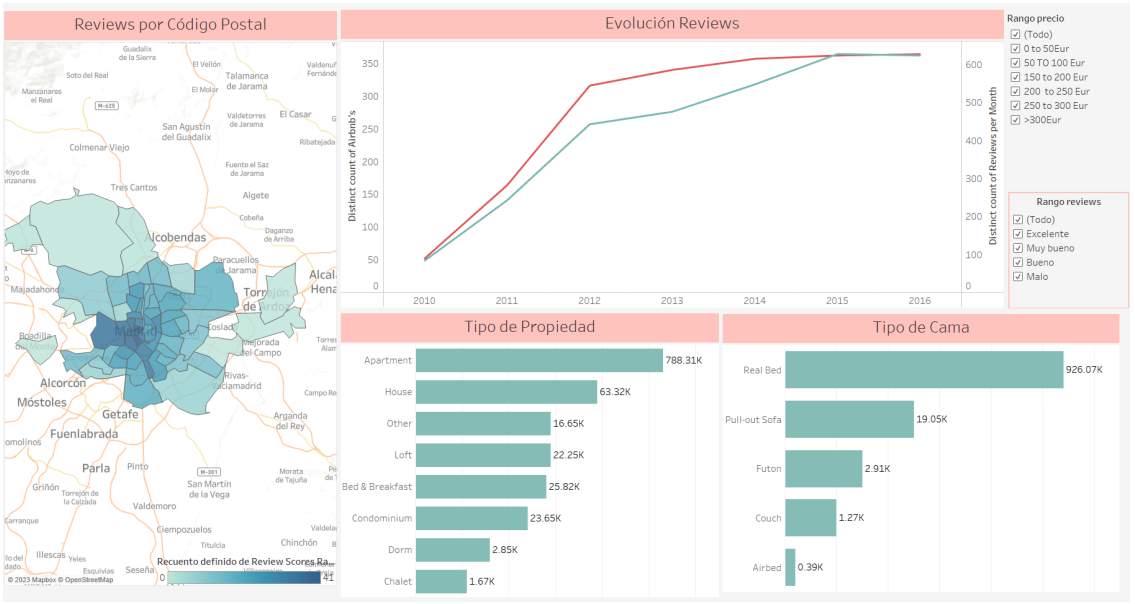
Después creamos el dashboard de las reviews:

Añadimos un mapa con el recuento único de reviews por código postal y a detalle de zona y barrio. El mapa también se puede usar para filtrar la información de las vistas de todo el dashboard.

Incluimos un gráfico de eje doble que muestra la evolución del recuento único de reviews y Airbnb's, vemos cómo a mayor número de Airbnb's hay mayor número de reviews.

Incluímos gráficos de barra con la información del número de reviews por tipo de propiedad y por tipo de cama.

Para este dashboard hemos creado un campo calculado en el que establecemos una escala de valoración en función del "Review Score Value": Rango reviews, que podemos usar como filtro.



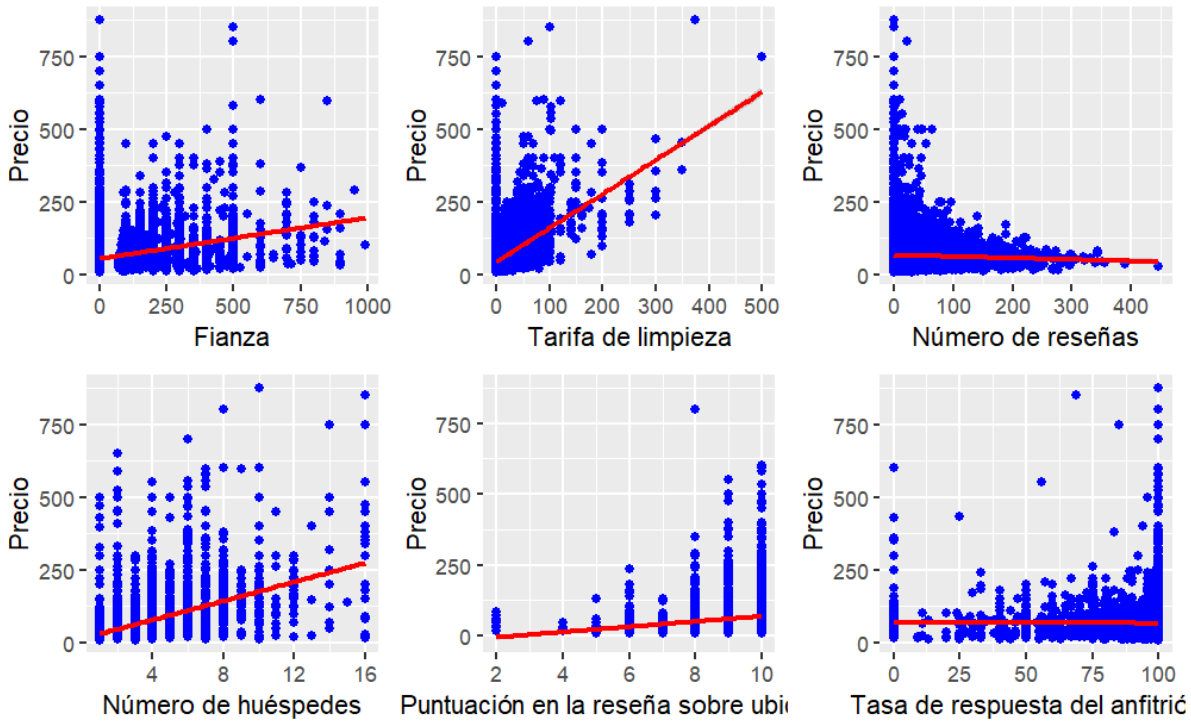
5. Creación del modelo de regresión lineal

El objetivo de este apartado es construir un modelo de regresión lineal que pueda predecir con cierta exactitud el precio de un inmueble en función de ciertas características.

Inicialmente, las características elegidas fueron las siguientes:

- fianza (Security.Deposit)
- tarifa de limpieza (Cleaning.Fee)
- número de reseñas (Number.of.Reviews)
- número de huéspedes (Accommodates)
- puntuación en la reseña sobre ubicación (Review.Scores.Location)
- tasa de respuesta del anfitrión (Host.Response.Rate)

En las siguientes gráficas podemos ver cómo se relacionan entre sí:



Como podemos apreciar, la evolución del precio del inmueble con respecto a las variables seleccionadas no siempre sigue una tendencia lineal.

Antes de realizar el modelo, dividimos los datos de partida, creando de esta forma un dataset de training y otro de testing.

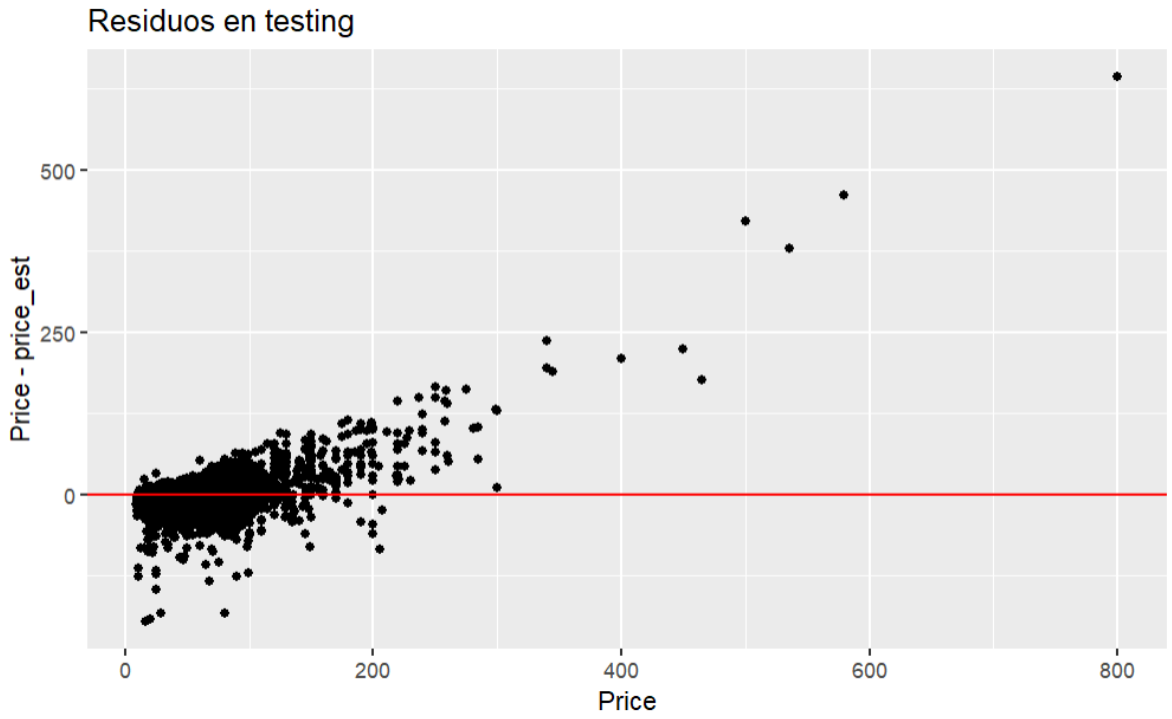
A continuación, hacemos el modelo para el dataset de training:

model <- lm(data=df.train, formula = Price ~ Security.Deposit+Cleaning.Fee+Number.of.Reviews+Accommodates+Review.Scores.Location+Host.Response.Rate)

El resultado obtenido es el siguiente:

*Price = -43.3 + 0.04*Security.Deposit + 0.67*Cleaning.Fee -0.056*Number.of.Reviews + 12.4*Accommodates + 6.47*Review.Scores.Location - 0.08*Host.Response.Rate*

Para este modelo, el valor del coeficiente de correlación R^2 es 0.54 para training, y 0.48 para testing. Aquí vemos que tenemos un R^2 que no es demasiado bueno. Comprobemos el residuo:



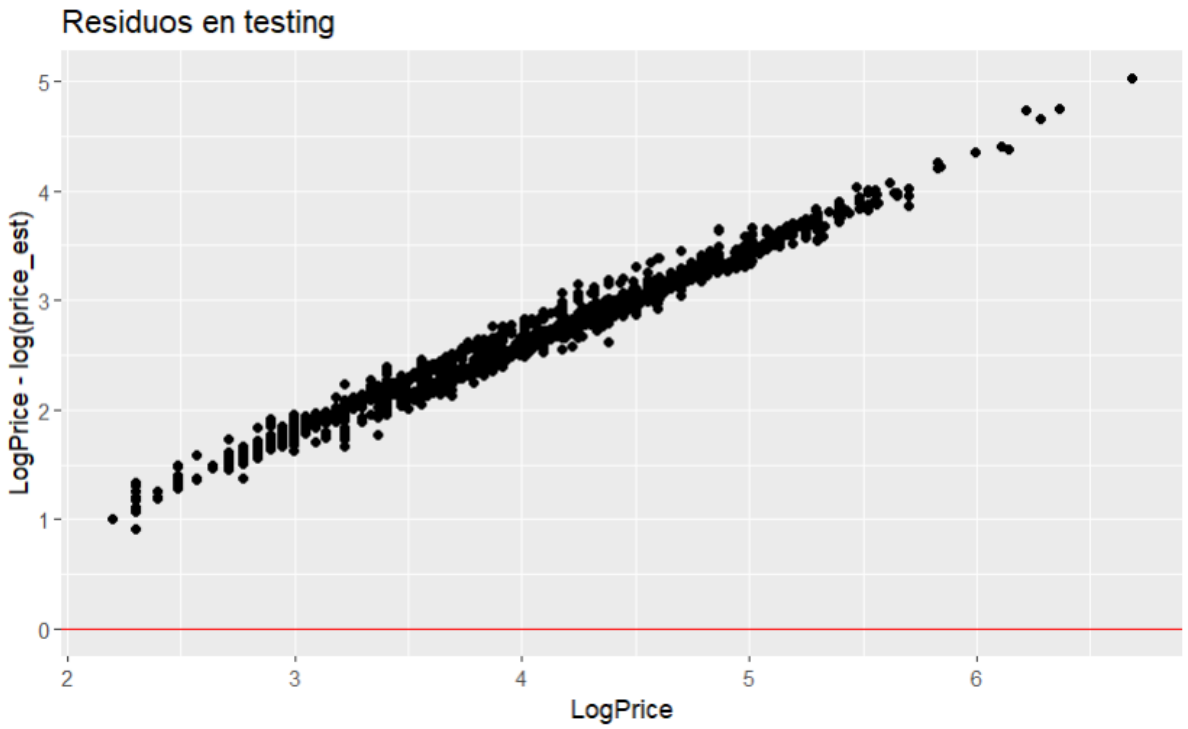
Se ve cómo a medida que aumenta el precio del inmueble, también lo hace el error del modelo. Este modelo no es especialmente bueno, pero podría ser útil si es rentable económicamente.

En cualquier caso, vamos a tratar de mejorarlo, añadiendo más variables como el número de habitaciones (Bedrooms), el número de baños (Bathrooms) y el tipo de habitación (Room Type), y usando la columna LogPrice que contiene los precios en escala logarítmica:

model <- lm(data=df.train, formula = LogPrice ~ Security.Deposit+Cleaning.Fee+Number.of.Reviews+Accommodates+Bedrooms+Bathrooms +Room.Type+Review.Scores.Location+Host.Response.Rate)

En este caso, el valor de R^2 es 0.72 para training, y 0.71 para testing, por lo que la exactitud del modelo ha mejorado considerablemente.

De nuevo, comprobamos el residuo:



Como vemos,el residuo es similar con respecto al modelo anterior.

3. Mejoras

La limitación de tiempo y a veces de conocimiento nos ha dejado con algunos puntos que podríamos mejorar.

1. **Rendimiento de la ETL**. El Notebook de Python encargado de la ETL va bastante lento, sobre todo porque en bastantes casos iteramos por todas las filas, que pueden rondar llegar a ser 14.000. Si hiciéramos otra vez el proyecto, una de las prioridades sería aligerar este proceso. Además, la inserción a la base de datos PostgreSQL sigue siendo algo manual, ya que requiere la ejecución del Script SQL que genera el Notebook. Con un poco más de tiempo y confianza en Python, habríamos abordado la librería 'SQLAlchemy' para automatizar verdaderamente el proceso de inserción de datos.
2. **Uso más amplio de variables.** Pese a que al principio seleccionamos 39 columnas que potencialmente podríamos usar, no acabamos de utilizarlas o analizarlas todas. Nos hubiese gustado especialmente usar 'features' y 'amenities' por el trabajo que costó limpiarlas, pero finalmente las dejamos fuera. En un futuro se podrían usar para enriquecer el Dashboard de tableau por ejemplo, o influir en el modelo predictivo.
3. **Modelo predictivo.** Pese a que solo se pedía un modelo predictivo para el precio, una posible mejora sería crear otros modelos algo más complejos que intenten reflejar mejor la realidad, ya que una simple regresión lineal se puede quedar corta y mostrar una visión parcial o simplificada de un tema tan complejo.

4. Conclusiones

En conclusión, este proyecto tenía como objetivo detectar las principales variables que afectan a los precios del alquiler de Airbnb en la ciudad de Madrid. Las hipótesis planteadas fueron: la ubicación del alojamiento y el número de habitaciones y camas; la renta per cápita del barrio; y el número de reviews positivas.

Los resultados obtenidos a través del análisis de los datos respaldan algunas de las hipótesis planteadas, en cambio hemos observado que otras no son relevantes.

En primer lugar, se ha demostrado que la ubicación del alojamiento es un factor determinante en el precio, siendo los alojamientos situados en los barrios céntricos y turísticos más caros que los situados en zonas menos populares. Además, el número de habitaciones y camas también influye en el precio, siendo los alojamientos con más habitaciones y camas más caros que los de menor tamaño.

En segundo lugar, se ha encontrado una relación entre renta per cápita del barrio y en el precio del alojamiento, siendo los alojamientos situados en barrios con una renta per cápita más alta más caros que los situados en barrios con una renta per cápita más baja.

Por último, se observa que las reviews positivas no tienen influencia en el precio del alojamiento, siendo los alojamientos con más reviews positivas más caros que los alojamientos con menos reviews.

Por otro lado, se han detectado otros elementos que están correlacionados con el precio de los alojamiento, como el cleaning fee. La tasa de limpieza aumenta a medida que lo hace el precio de alquiler.

A la hora de realizar la regresión lineal ha sido necesario incluir más variables que las iniciales, como el número de baños, el tipo de habitación y los precios en escala logarítmica para lograr un modelo más fiable.