PRÁCTICA FINAL

BOOTCAMP MUJERES EN TECH

KEEPCODING — GLOVO

DATASOUTH

Claudia Aldana Bilbao
Irene Pozo Parejo
Lourdes Martínez Martínez
Virginia Ordoño Bernier

15 febrero 2023

Índice de contenidos

Índice de figuras

1 Introducción

Esta práctica se va a llevar a cabo a partir de un archivo .csv cuyos datos han sido proporcionados por la empresa Airbnb. Partiendo de estos datos se realizará un filtrado por la ciudad de Madrid y otros aspectos que se verán a lo largo del documento.

1.1. Objetivo

El objetivo principal es realizar un estudio de mercado para establecer la relación entre los alojamientos y su precio en los distintos distritos de la ciudad de Madrid. Se parte de la suposición inicial de que los alojamientos más céntricos de Madrid se corresponderían con aquellos cuyo precio sería más elevado. Para ello, se han tenido en cuenta distintos parámetros como son las características y los servicios de dichos alojamientos, ya que se ha considerado que pueden ser importantes para la consecución de este objetivo.

Desarrollo 2

2.1. Recursos y herramientas

Los recursos y herramientas utilizados durante el desarrollo del proyecto son los siguientes:

- **Arquitectura y validación de datos**: Google Colab (*pandas*, *psycopg2*), PostgresSQL y ElephantSQL
- **Análisis exploratorio**: Google Colab (*pandas*, *re*, *matplotlib.pyplot*)
- **Visualización de métricas**: Tableau
- **Pre-procesamiento y modelado**: RStudio (*tidyverse*, *ggplot2*, *GGally*)

2.2. Análisis exploratorio

2.2.1. Definición del dataset

Inicialmente, se realiza un filtro para quedarse con las columnas con las que se va a trabajar a partir de ahora. Estas son las siguientes: 'ID', 'Neighbourhood Group Cleansed', 'City', 'Latitude', 'Longitude', 'Zipcode', 'Property Type', 'Room Type', 'Bed Type', 'Beds', 'Bedrooms', 'Bathrooms', 'Accommodates', 'Amenities', 'Price', 'Security Deposit', 'Cleaning Fee', 'Cancellation Policy', 'Review Scores Rating' y 'Market'.

2.2.2. Análisis exploratorio con Python

Primero, se procede a realizar un filtro para quedarse con los datos de la ciudad de estudio, en este caso Madrid.

Se ha contabilizado cuántos alojamientos hay de cada tipo ('Property Type') y como más del 80 % de las propiedades son de tipo 'Apartment', se van a estudiar los objetivos sobre estos datos. También, como se ha comentado al definir el objetivo, se quiere comparar el precio de las distintas propiedades para los distritos de Madrid y según las características de dichas propiedades. Para ello, primero se estudian los tipos de 'Room Type' que se tienen en los distintos apartamentos. Se obtienen tres resultados:

- "Entire home/apt" (apartamento completo)
- "Private room" (habitación privada)
- "Shared room" (habitación compartida)

El estudio sobre 'Room Type' se centrará en el apartamento completo y en la habitación privada, ya que la habitación compartida constituye menos del 1 % de los datos.

Después de un estudio previo del dataset se han visto algunas columnas en las que hay valores vacíos y se ha decidido darles un valor por defecto de la forma que se considera más conveniente a las columnas de interés anteriormente mencionadas; ya sea con el valor 0 para las columnas numéricas, un string vacío ' ' para las columnas alfanuméricas o la eliminación de las filas en las que los datos no puedan ser estimados. También se realiza una conversión de divisa de $ a €.

A continuación, se hace un tratamiento al contenido de la columna 'Amenities', obteniendo un booleano como respuesta, que constituye la creación de unas nuevas columnas donde se indica si el apartamento dispone de cada amenitie o no. Se han elegido las siguientes: 'Internet', 'TV', 'Kitchen', 'Air conditioning' y 'Heating', ya que se les ha considerado las más relevantes.

Seguidamente, se cambia el nombre de la columna 'Neighbourhood Group Cleansed' por 'Neighbourhood', se eliminan los espacios de los títulos de las columnas y se cambian los valores de la columna 'Review Scores Rating' a enteros.

El siguiente paso que se sigue es detectar los outliers (valores atípicos) del dataset final. Esto se hace porque con estos valores se obtienen unos histogramas cuyos valores son singulares, lo que se puede observar en las siguientes gráficas, para las que se explican sus valores.

En la Figura 2.1 se pueden observar los histogramas de las columnas que tienen más relevancia a la hora del estudio de outliers. Examinando estos gráficos, se estima que los valores obtenidos para 'Beds', 'Bedrooms, 'Bathrooms' y 'Accomodates' tendrán poca diferencia eliminando dichos valores atípicos, por lo que para estas columnas se mantendrán los outliers. Por el contrario, en el caso de los diferentes precios ('Price', 'SecurityDeposit' y 'CleaningFee') se contempla que los valores pueden ser algo elevados para dichos alojamientos, por lo que se graficarán sus correspondientes *boxplots* para valorar si se eliminan los outliers.

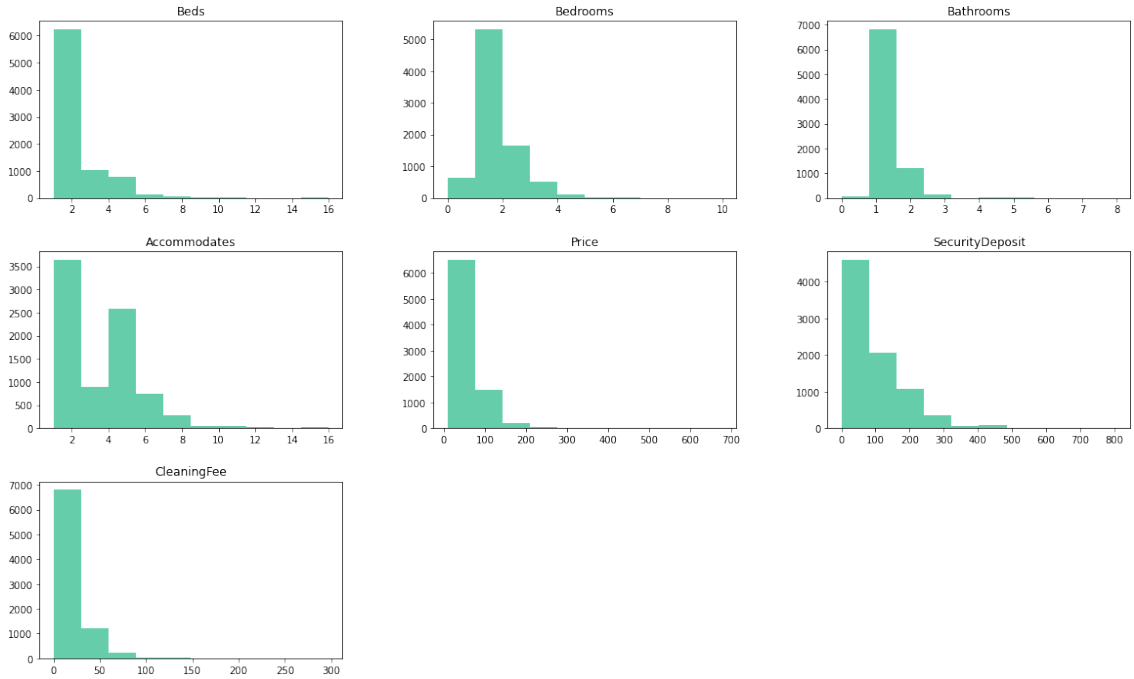
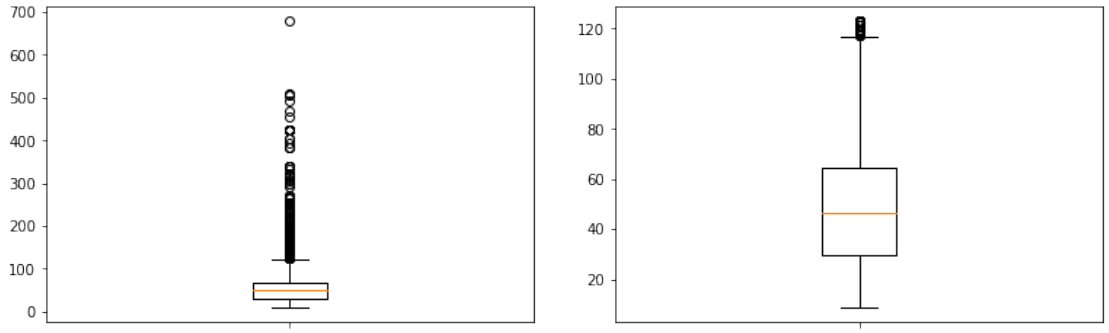


Figura 2.1: Histograma de los datos con outliers

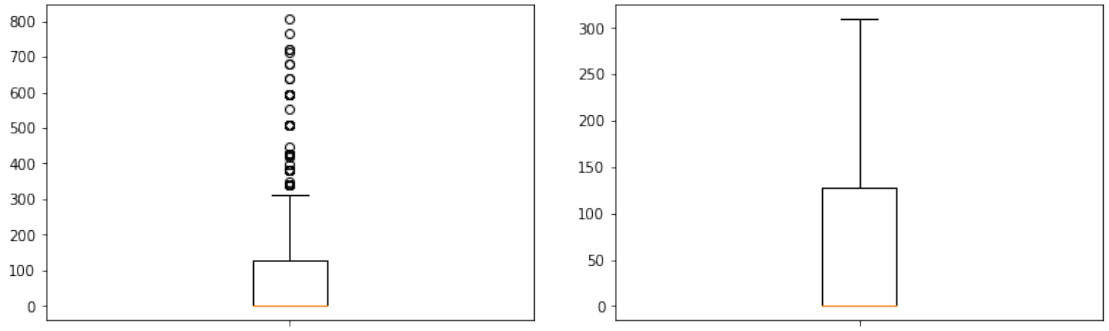
Para poder detectar los outliers se ha utilizado el rango intercuartílico (IQR), que se define como la diferencia entre el percentil 75 ($Q3$) y el percentil 25 ($Q1$) en un grupo de datos. Un dato se considera un outlier si es 1,5 veces el rango intercuartílico mayor que el tercer cuartil ($Q3$) o 1,5 veces el rango intercuartílico menor que el primer cuartil ($Q1$). Luego de esta predicción de outliers, se procede a su eliminación en las columnas 'Price', 'SecurityDeposit' y 'CleaningFee'.



(a) *Boxplot* con outliers **(b)** *Boxplot* sin outliers

Figura 2.2: *Boxplots* del precio

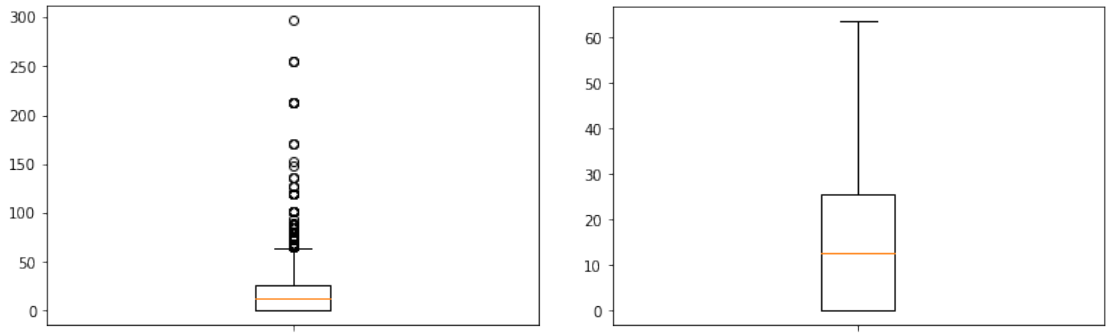
Como se puede ver en la Figura 2.2(a) los valores que salen con outliers son muy elevados, además de que hay muchos valores atípicos. Una vez eliminados la mayoría de dichos valores, el rango de precio es algo más razonable, como se ve en la Figura 2.2(b). Aunque estos valores se hayan eliminado, esto no ha afectado a la mediana, ya que sigue estando más o menos en el medio entre los cuartiles 1 ($Q1$) y 3 ($Q3$), por lo que la distribución de los datos es simétrica.



(a) *Boxplot* con outliers **(b)** *Boxplot* sin outliers

Figura 2.3: *Boxplots* de la fianza

En el caso de la fianza, también se observa que los valores del precio con outliers son bastante elevados (Figura 2.3(a)) respecto a los valores obtenidos una vez que se han eliminado los valores atípicos (Figura 2.3(b)). En este caso, la mediana coincide con el cuartil 1 ($Q1$), esto significa que la distribución de datos es asimétrica.



(a) *Boxplot* con outliers **(b)** *Boxplot* sin outliers

Figura 2.4: *Boxplots* de la tasa de limpieza

Por último, se muestra en la Figura 2.4 los *boxplots* de la tasa de limpieza con y sin outliers. Con los outliers el valor del precio de la tasa de limpieza se dispara, además de que en el gráfico se muestran la cantidad de valores atípicos que existen. Una vez eliminados dichos valores, el rango de precio de la tasa de limpieza es más razonable ya que interesan aquellos alojamientos cuyo rango de precio esté entre 0 y 100 €.

Una vez hecho todo lo anterior, se exporta el archivo .csv para su posterior uso con el software Tableau.

Cuando se estudiaron los datos, en la columna 'Zipcode' había datos de códigos postales que no pertenecían a la zona o que no servirían en este caso. Para su eliminación se utiliza una expresión regular (RegEx) gracias a la cual se filtran los datos para finalmente mantener solo los códigos postales entre 28001 y 28991. Luego de esto, se exporta el archhivo .csv para su posterior uso con el software RStudio.

Tanto para el alojamiento completo como para la habitación privada, se ha calculado el precio total medio por cada distrito (utilizando la función *groupby()* y la

media). Una vez hecho esto, como es lógico se obtiene que el precio total medio es más alto para el alojamiento completo que para la habitación privada, pero en cambio los distritos más caros para un alojamiento completo (Barajas, Salamanca, Retiro, Centro y Moncloa-Aravaca) no coinciden completamente con los distritos más caros para una habitación privada (Centro, Chamberí, Chamartín, Barajas y Tetuán).

Para graficar los distritos y los tipos de habitación frente el precio, se calcula el precio medio (mediano en este caso, ya que da una idea más fiable de lo que ocurre en la realidad) por barrio y tipo de habitación.

En el caso de la gráfica de los diferentes distritos frente al precio, se puede observar que tanto el distrito Centro como el distrito Chamartín comparten el valor más elevado, seguido de los distritos de Salamanca y Retiro. Por otra parte, los distritos más baratos son la Latina y Usera, como se observa en la Figura 2.5.

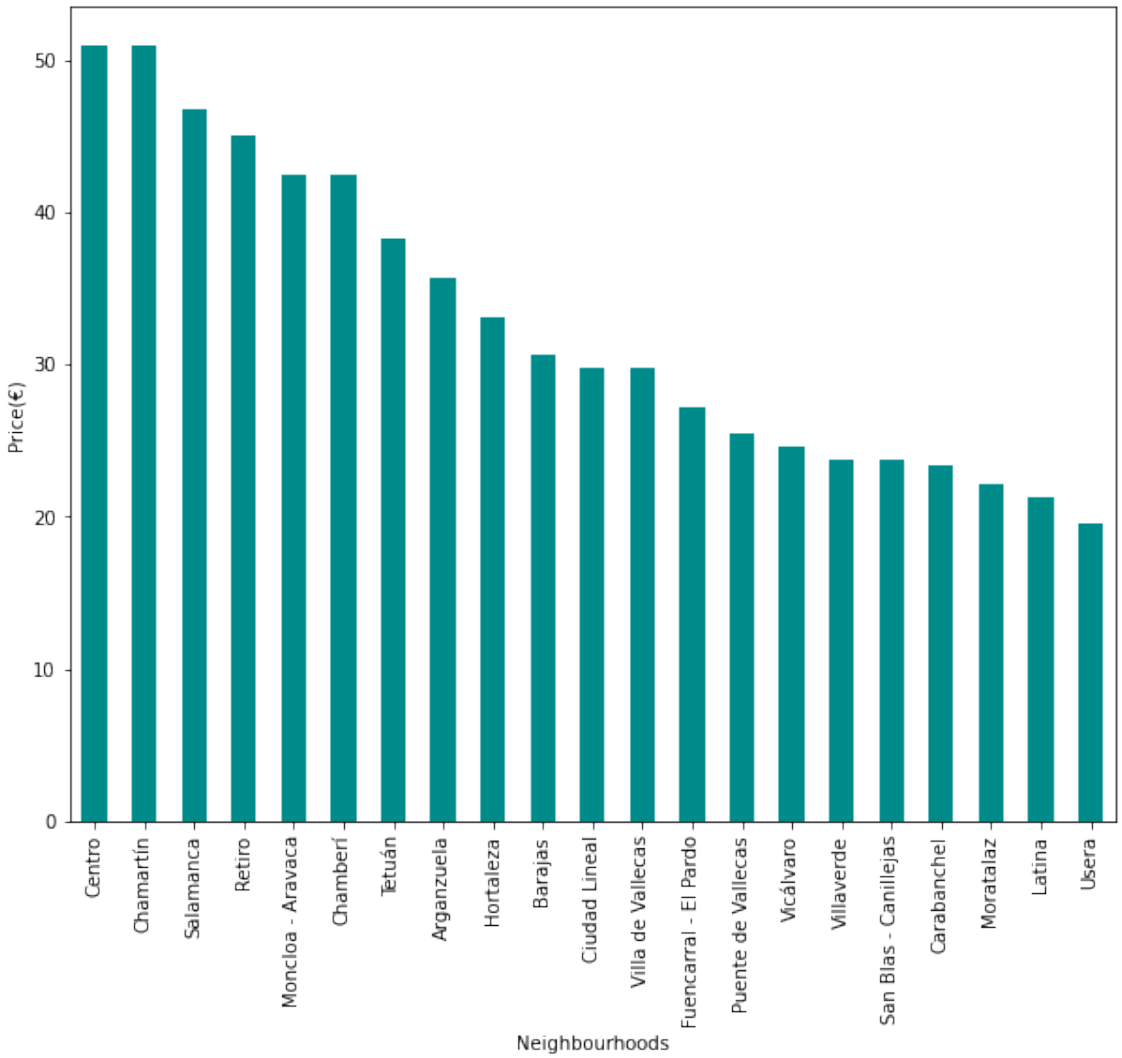


Figura 2.5: Precio medio según el distrito

Por su parte, se aprecia en la Figura 2.6 que el precio medio de los apartamentos completos es más del doble que el de las habitaciones privadas en apartamentos compartidos, como es lógico.

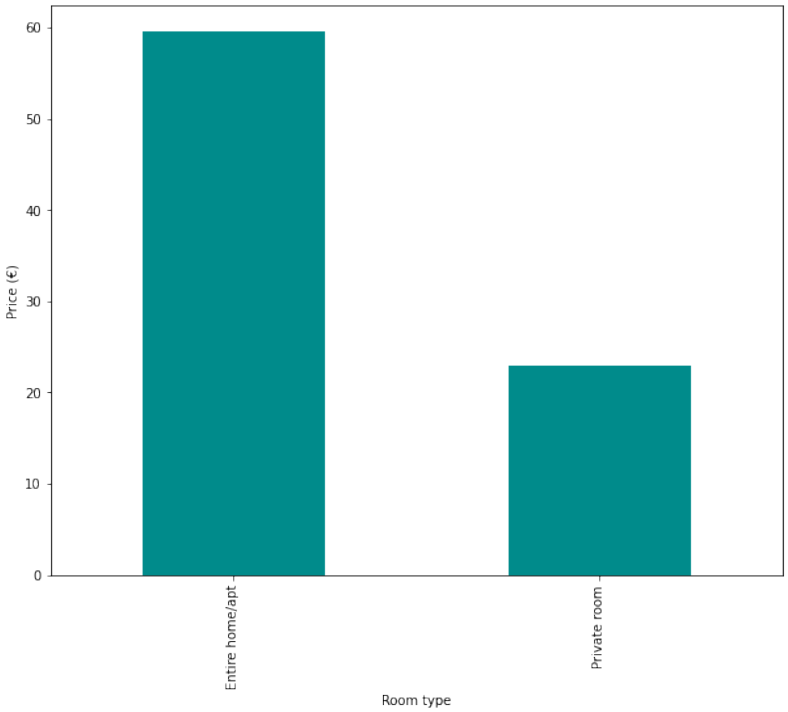


Figura 2.6: Precio medio según el régimen de alojamiento

2.3. Arquitectura y validación de los datos

2.3.1. Conexión y modelado de datos

Todo este proceso se ha dividido en 4 scripts en Google Colab:

2.3.1.1. Script 1: Preparación del entorno

Se importan las librerías *pandas* (estructura de datos y herramientas de análisis de datos) y *psycopg2* (conexión y manipulación de la base de datos PostgreSQL) de Python. Se usa ElephantSQL como servicio de alojamiento de base de datos PostgresSQL. A continuación, se carga el archivo .csv creado con los datos ya filtrados obtenidos del dataset original.

2.3.1.2. Script 2: Conexión y modelado de datos

Debido al tiempo que se demora la carga de los datos y para facilitar todo el proceso el día de la presentación del proyecto, toda la estructura de la base de datos se encuentra creada, al igual que los datos insertados.

1. **Declaración de variables**:
   - *restart = False*. Al cambiar su valor a *True* permite poder realizar todo el proceso desde el inicio si así se desea.
   - *cur = None*. Representa el cursor usado para las consultas.
   - *con = None*. Representa la conexión a la base de datos.

2. **Bloque *try/catch***: tiene como objetivo asegurarse de que la conexión a la base de datos se cierre siempre, independientemente de si el código dentro del bloque *try* se ejecuta correctamente o no. De esta forma, se evita que, en caso de producirse una excepción, la conexión a la base de datos pueda quedar abierta y consumir recursos de manera innecesaria. En el bloque *finally*, se cierra la conexión a la base de datos si aún existe, verificando si *cur* y *conn* no son *None* antes de cerrarlos y evitando así que al intentar cerrarla se pueda producir un error. Dentro del bloque *try*, se encuentra todo el proceso de conexión y modelado que ahora se va a detallar.

   Conexión y creación del esquema

   a) Delimitación de la columna 'Amenities' en el dataframe a 255 caracteres para evitar problemas de longitud de inserción, tras comprobar que la última parte de la columna no es relevante para el tratamiento de los datos.

   b) Cambio del nombre de la columna 'ID' a 'IdAirbnb' ya que se usará un id propio y autoincrementable al crear la tabla. Se mantiene este campo como identificador del registro en el dataset original.

   c) Conexión con la base de datos en Elephantsql.

   d) Creación cursor para ejecutar consultas.

   e) Eliminación de *table* y creación de *schema*. Para este tipo de acciones en general, se usa siempre el *if exists* para evitar la generación de errores en caso de que no exista ese elemento en la base de datos.

   Modelado

   Se ha partido de la idea de no saber ni el número de columnas o filas, ni el nombre de las columnas, ni el tipo de datos de cada una de ellas. Por lo tanto, se ha implementado un modelo que sirve para diferentes tipos de datasets, flexibilizando así el proceso de gestión de datos inicial.

   - Creación de una lista con los nombres de las columnas y otra con los tipos de datos de las columnas en el dataframe. Se usará más adelante.
   - Se recorre el dataframe y se detecta el tipo de dato de cada columna. Cada dato se transforma en su equivalente de tipo de dato de PostgreSQL y se insertan los valores en una lista.
   - Creación de una cadena con la definición de la tabla, agregando un nuevo campo id como clave primaria que sea autoincremental y que no acepte valores nulos. Se crea la tabla y se crea un string de la inserción con el nombre de las columnas.

- Por último, se inicializa una lista para almacenar las consultas SQL y se itera sobre las filas del dataframe. Se crea una consulta SQL para cada fila y se ejecuta.

2.3.1.3. Script 3: CRUD completo

Se ha realizado un CRUD completo de un registro inventado. Los pasos son:

- Inserción del nuevo registro.
- Obtención del *id* del último registro insertado con *lastval()*.
- Actualización del campo 'Beds'.
- Eliminación del registro y comprobación.

2.3.1.4. Script 4: Consultas

Las consultas que se han realizado son las siguientes:

- Distritos con tipo de propiedad en Madrid.
- Cantidad de elementos con 'Review Scores Rating' mayor del 95 %.
- Tipos de política de cancelación y cantidad de pisos por cada uno de ellos, ordenados de mayor a menor.
- Creación de una nueva columna con el cambio de divisa de dólares a euros.

2.4. Visualización de las métricas

2.4.1. Visualización de las métricas con Tableau

En este paso, primero se carga el dataset anteriormente creado. Como en dicho dataset ya estaba filtrado 'Market' por Madrid, 'PropertyType' por 'Apartment' y para 'RoomType' se ha excluido 'Shared Room', se crea un filtro para 'Zipcode'. Para ello, se añade una fórmula en la pestaña *Condición*, en la que se establece el rango de los códigos postales entre 28001 y 28991. La fórmula utilizada es la siguiente:

```
IF [Zipcode] <= 28991 AND [Zipcode] >= 28001
THEN TRUE
ELSE FALSE
END
```

A continuación, se crean los siguientes campos calculados:

- **Longitud y latitud float**. Con la función `FLOAT` se crean unos nuevos valores de longitud y latitud y se debe cambiar también la función geográfica una vez creados.
- **Precio final**. Este precio es el resultado de la suma de los promedios de 'Price', 'SecurityDeposit' y 'CleaningFee'. A estos valores también se les ha aplicado la función `FLOAT` ya que al crear el .csv los valores eran strings.
- **Fianza**. Se ha calculado también con la función `FLOAT` el depósito o fianza.
- **Reviews float**. Con la función `FLOAT` también se han sacado los porcentajes de las reviews.

Una vez hecho esto, primero se crea el gráfico **"Precio final promedio por distrito"**. El fitro que se ha añadido es el de precio final, ya que así se puede observar que conforme va disminuyendo el precio los barrios son más baratos, desapareciendo los más caros. El precio final también se ha añadido en color, para que esta variación de precios se pueda visualizar mejor. Además, se ha añadido un gráfico en forma de tarta (*pie chart*) en el que se muestran por cada distrito los tipos de 'RoomType'. Esto puede observarse en la Figura 2.7.

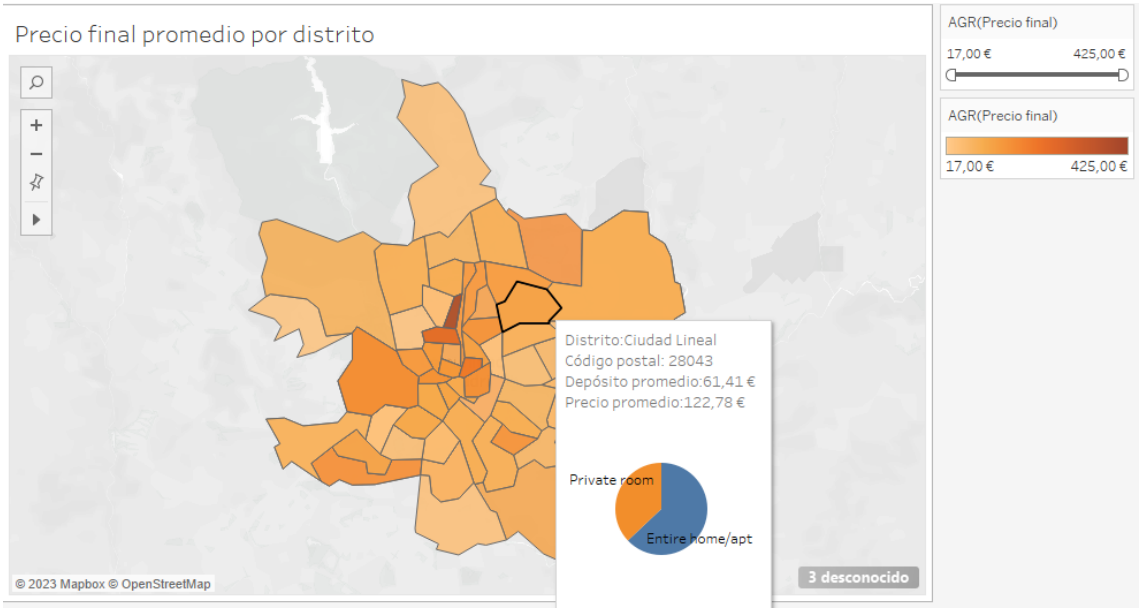


Figura 2.7: Precio final promedio por distrito

El siguiente gráfico que se ha realizado es **"Top N valoraciones por distrito"**. En este caso se trata de un diagrama de barras en el que se representa el precio final frente a los distritos, estando las reviews en color y etiqueta. Para este gráfico se ha creado el parámetro *Top N*, por defecto de 5 y se le añade al filtro de 'Neighbourhood'. Véase en la Figura 2.8.

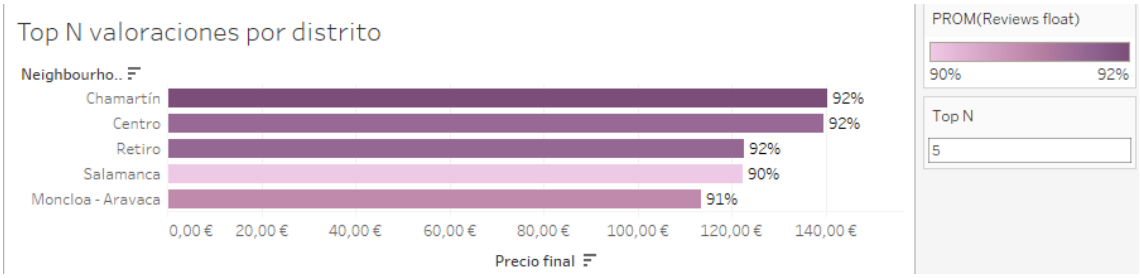


Figura 2.8: Top N valoraciones por distrito

Por último, se crea el gráfico **"Amenities por distritro"**. Para este gráfico se añaden las 'Amenities' que antes se comentaron a filtros para que se puedan seleccionar como un *radio button*. Este gráfico también es un mapa en el que se ha aplicado el filtro de colores por precio.

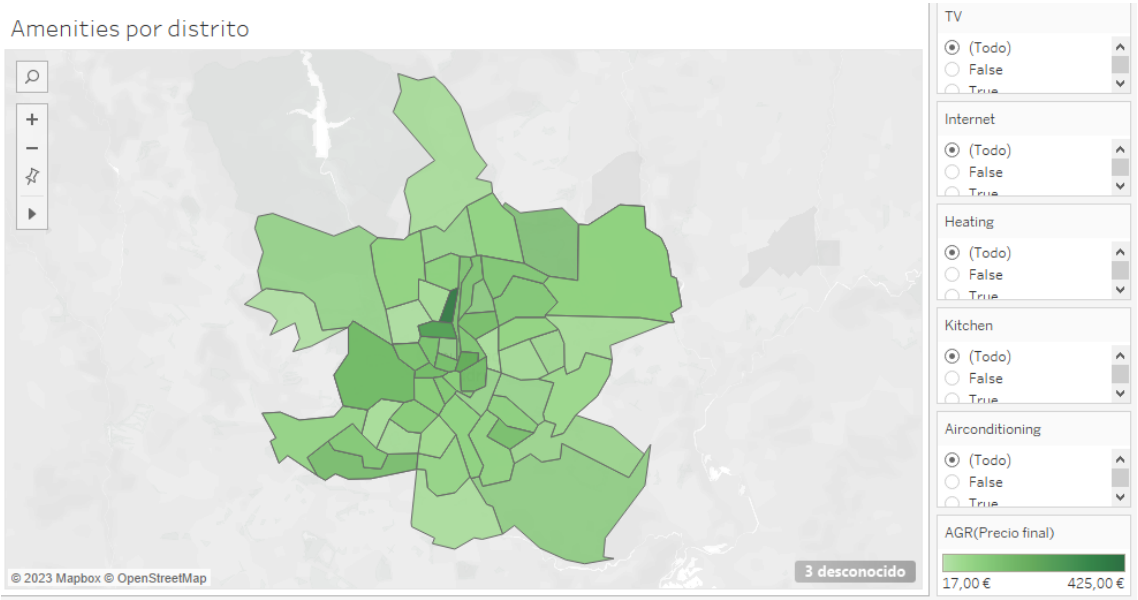


Figura 2.9: Amenities por distrito

Una vez están creados todos los gráficos, se procede a la creación del dashboard. Como se observa en la Figura 2.10 se ha variado el precio final del gráfico *Precio final promedio por distrito* y se ha filtrado por las diferentes 'Amenities', dando como resultado diferentes mapas y leyendas del precio final.

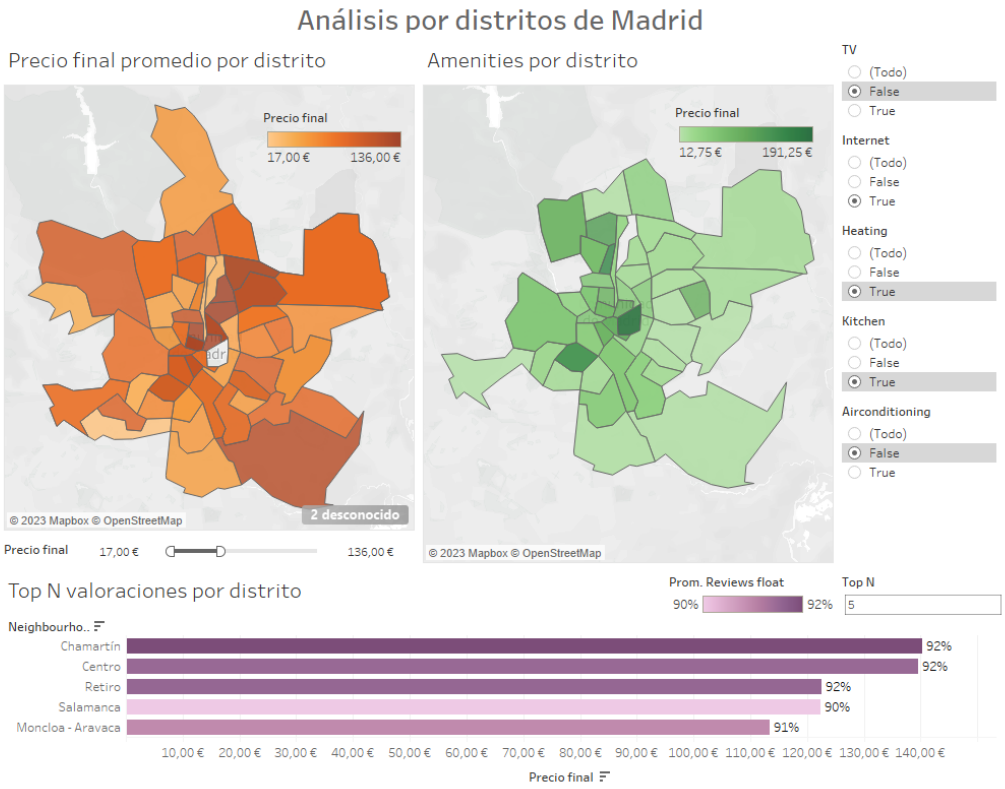


Figura 2.10: Dashboard

2.5. Pre-procesamiento y modelado

Para este objetivo, se selecciona un modelo de regresión lineal múltiple, ya que permite predecir el valor de la variable dependiente y (en este caso el precio), en función de las variables independientes (predictores) de interés.

2.5.1. Pre-procesamiento y modelado en Rstudio

Se importan las librerías necesarias (*tidyverse* y *GGally*) y el dataset generado previamente en Python.

Respecto a los predictores, se seleccionaron para generar el modelo: el distrito, tipo de habitación, número de huéspedes, camas, habitaciones, baños, tasa de limpieza, fianza, política de cancelación y puntuaje de la reviews. De este modo, se crearía un modelo que predijera el precio de los apartamentos en función de estos parámetros.

A continuación, se muestra la correlación entre las variables numéricas y el precio.

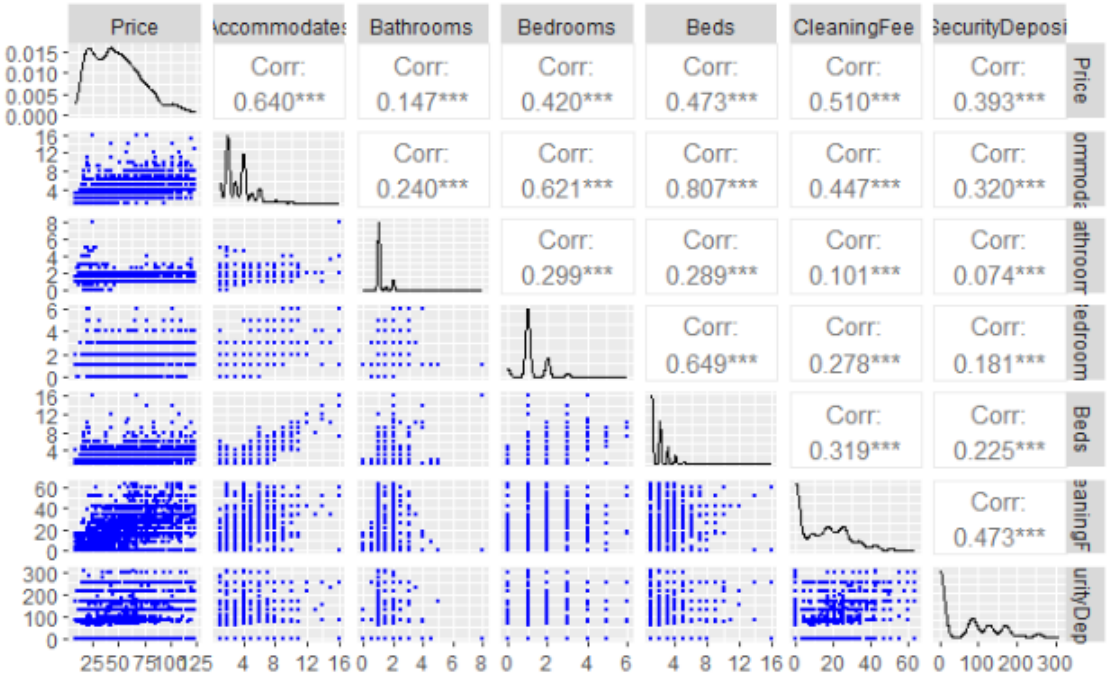


Figura 2.11: Correlaciones de las variables

Del análisis preliminar de correlación, a priori se han extraído las siguientes conclusiones:

- En general, ninguna de las variables tienen una correlación de 1, pudiéndose suponer los predictores como independientes, por lo que no existe colinealidad en el modelo, aunque habría que comprobarlo por regresión lineal simple de cada uno de los predictores.
- El precio tiene una mayor correlación con el número de huéspedes en primer lugar, así como con el número de habitaciones, número de camas y la tasa de limpieza, por lo que estas variables pueden ser importantes en el modelo.
- La correlación más elevada la presenta el número de huéspedes, que parece estar más correlacionado con el número de camas y de habitaciones. Por el contrario, el puntuaje de las reviews es el que menor correlación muestra con el resto de variables.

2.5.2. Generación del modelo

Se han dividido los datos en dos grupos: **train** y **test**. El grupo train contiene el 70 % de los datos y sirve para entrenar el modelo. El grupo test sirve para evaluar la calidad del modelo.

Primero, se calcula el modelo con el dataset train, mediante la función *lm()*.

```
Call:
lm(formula = Price ~ Neighbourhood + RoomType + Accommodates +
    Bathrooms + Bedrooms + CleaningFee + SecurityDeposit + CancellationPolicy,
    data = df_out.train)

Residuals:
    Min      1Q  Median      3Q     Max
-78.519  -9.480  -1.417   7.223  67.854

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                              28.328434   1.124399  25.194  < 2e-16 ***
Neighbourhood[T.Barajas]                 -0.965744   3.020472  -0.320 0.749183
Neighbourhood[T.Carabanchel]             -5.495537   1.602277  -3.430 0.000609 ***
Neighbourhood[T.Centro]                   6.673550   0.869865   7.672 2.00e-14 ***
Neighbourhood[T.Chamartín]                7.365297   1.546154   4.764 1.95e-06 ***
Neighbourhood[T.Chamberí]                 4.651724   1.150936   4.042 5.38e-05 ***
Neighbourhood[T.Ciudad Lineal]           -4.194470   1.702485  -2.464 0.013781 *
Neighbourhood[T.Fuencarral - El Pardo]   -0.503826   2.523897  -0.200 0.841784
Neighbourhood[T.Hortaleza]                3.489931   2.230955   1.564 0.117802
Neighbourhood[T.Latina]                  -6.621799   1.479114  -4.477 7.73e-06 ***
Neighbourhood[T.Moncloa - Aravaca]        5.052047   1.554472   3.250 0.001161 **
Neighbourhood[T.Moratalaz]               -6.558344   2.718623  -2.412 0.015883 *
Neighbourhood[T.Puente de Vallecas]      -7.981650   1.906840  -4.186 2.89e-05 ***
Neighbourhood[T.Retiro]                   5.223298   1.409526   3.706 0.000213 ***
Neighbourhood[T.Salamanca]                6.073027   1.208435   5.026 5.18e-07 ***
Neighbourhood[T.San Blas - Canillejas]   -1.862110   2.680384  -0.695 0.487263
Neighbourhood[T.Tetuán]                  -0.854418   1.430140  -0.597 0.550241
Neighbourhood[T.Usera]                   -5.725783   2.197986  -2.605 0.009213 **
Neighbourhood[T.Vicálvaro]               -7.323887   6.229545  -1.176 0.239780
Neighbourhood[T.Villa de Vallecas]        0.703822   4.268235   0.165 0.869031
Neighbourhood[T.Villaverde]              -7.606207   3.397943  -2.238 0.025231 *
RoomType[T.Private room]                -22.820298   0.607370 -37.572  < 2e-16 ***
Accommodates                              3.152502   0.195005  16.166  < 2e-16 ***
Bathrooms                                 3.620517   0.509486   7.106 1.35e-12 ***
Bedrooms                                  6.218890   0.403563  15.410  < 2e-16 ***
CleaningFee                               0.148234   0.019495   7.604 3.38e-14 ***
SecurityDeposit                           0.021461   0.003139   6.838 8.96e-12 ***
CancellationPolicy[T.moderate]           -1.409393   0.552299  -2.552 0.010742 *
CancellationPolicy[T.strict]             -1.558742   0.552951  -2.819 0.004836 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.11 on 5316 degrees of freedom
Multiple R-squared:  0.6291,    Adjusted R-squared:  0.6271
F-statistic:   322 on 28 and 5316 DF,  p-value: < 2.2e-16
```

Figura 2.12: Datos del modelo de regresión lineal

El modelo tiene un R^2 (0.63 aprox.) que es capaz de explicar el 63 % de la variabilidad observada en el precio. El p-value del modelo es significativo $(2{,}2e-16)$ por lo que se puede aceptar que el modelo no se genera por azar.

Las variables que más influencia tienen en el modelo son las que presentan un p-valor (Pr) más bajo. Además, el grado de confianza de las variables en el modelo indica que el número de camas ('Beds') no tiene apenas influencia en el modelo, es decir, el precio del alquiler no va a depender del número de camas cuando se utilizan estos predictores. Este mismo hecho se ha presentado en alguno de los distritos, que muestran un p-valor más elevado pero un menor grado de confianza. Algunos de estos son los distritos de Barajas, Fuencarral-El Pardo, San Blas-Canillejas, Tetuán, Vicálvaro y Villa de Vallecas. Estos distritos, exceptuando Tetuán, coinciden con

antiguos municipios absorbidos por la ciudad, por lo que son distritos situados más lejos del centro.

2.5.3. Calidad del modelo

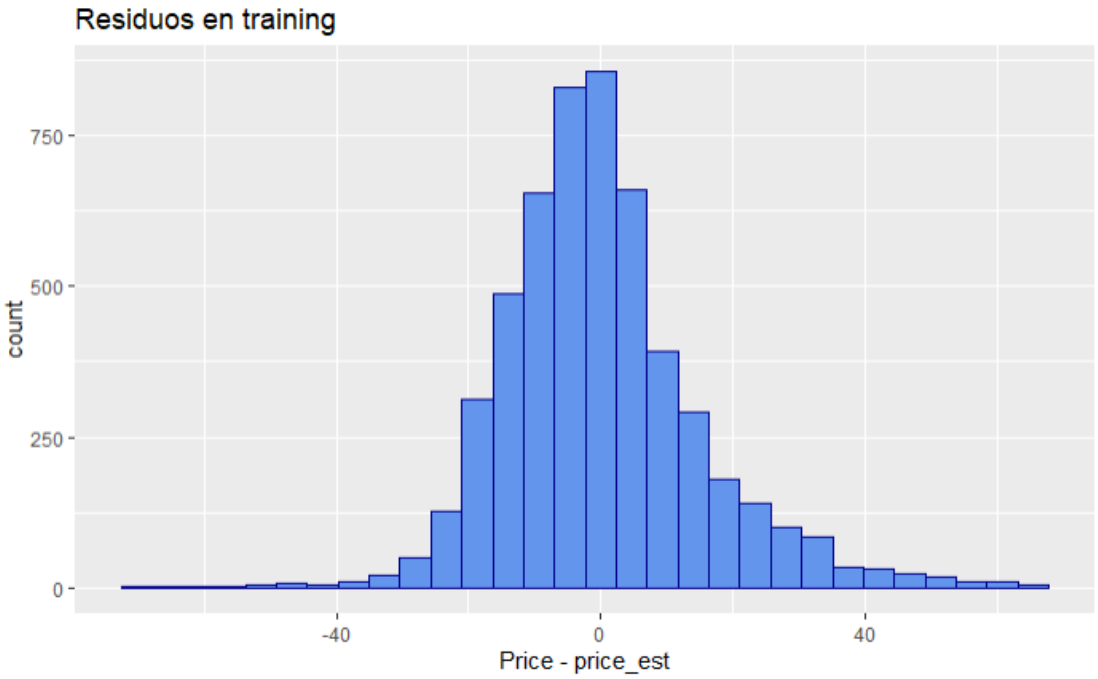
Se han calcualado las figuras de calidad del modelo, tanto para el grupo de training como para el grupo de testing.

```
      RMSE   Rsquared        MAE
15.0648250  0.6290958 11.2118688
      RMSE   Rsquared        MAE
14.8138111  0.6375004 10.9442627
```

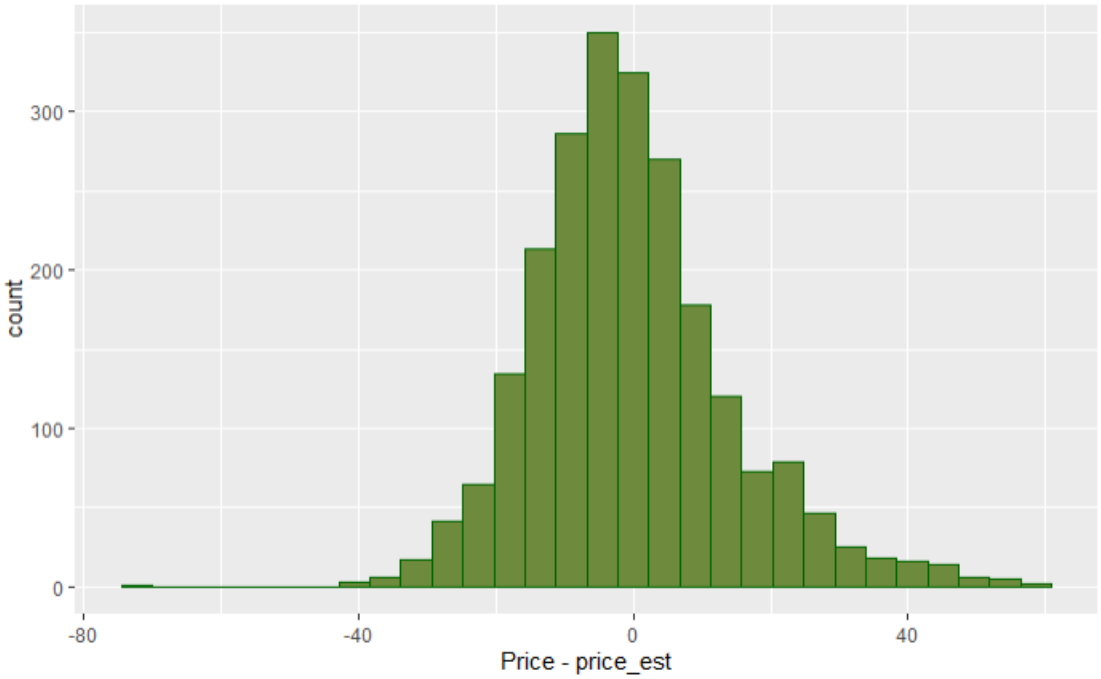
Figura 2.13: Parámetros de calidad de los datos en train y test.

El R^2 del modelo es próximo a 0.64 en ambos modelos de train y test. Además sus medidas de RMSE y R^2 son muy parecidas, lo que indica que no hay sobreajuste en el modelo.

Por otro lado, se han estudiado los residuos del grupo training (Figura 2.14(a)) y testing (Figura 2.14), para asegurar que seguían una distribución normal, con una varianza constante y unos valores medios centrados en 0.



(a) Training

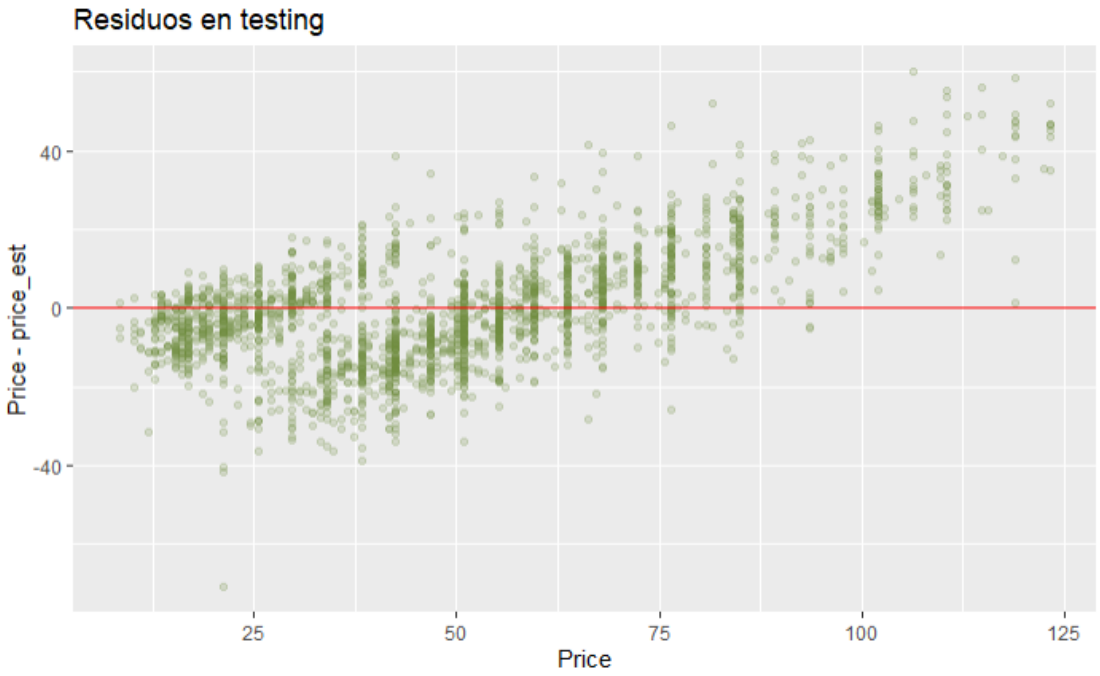


(b) Testing

Figura 2.14: Histogramas de calidad

Como se puede observar en la Figura 2.15, los residuos siguen una distribución normal tanto en el grupo train como en el test. Sin embargo, a medida que aumenta el precio también aumenta el error del modelo. Observando la distribución de los residuos, el modelo da unos valores ligeramente pesimistas conforme se aumenta el precio, y optimistas cuando los valores del precio son muy pequeños.

(a) Training



(b) Testing

Figura 2.15: Gráficas de residuos

Además, se han extraído los quantiles, que indicaban que el 95 % de las muestras se encuentran en valores entre -25 y 35 aproximadamente.

Este modelo no predice especialmente bien en aquellos valores extremos, con un precio muy elevado, pero podría ser útil si es rentable, sobre todo en los valores más centrales. Además, ha proporcionado información sobre aquellas variables que no influyen en el precio de los apartamentos, como son el número de camas o la pertenencia del inmueble a algunos distritos.

3 Conclusiones

Este proyecto parte de la **suposición inicial** de que los alojamientos más céntricos tendrán un precio más elevado. Gracias al análisis exploratorio se puede conseguir un tratamiento óptimo de los datos, excluyendo aquellos no relevantes. Además, se puede observar que los barrios más céntricos se corresponden con los barrios más caros, teniendo en cuenta únicamente el precio del alquiler y el distrito. Sin embargo, el usuario también puede seleccionar otras característicasa la hora de alquilar el inmueble, como por ejemplo la política de cancelación o la puntuación de las reseñas así como los servicios ofrecidos.

Por ello se procedió en Tableau a observar la distribución de los precios de los alojamientos. Siendo esta distribución más heterogénea, es decir, los alojamientos de mayor precio no se sitúan en los barrios más céntricos, sino que dependen además de otros parámetros como son la fianza y la tasa de limpieza. De la misma manera ocurre para los servicios ofrecidos por los alojamientos. Por otra parte, se deduce que la puntuación de las reseñas no necesariamente se corresponde con los apartamentos más caros.

Ya que el precio del inmueble depende de varios factores como su localización, los servicios y las características, se realiza un modelo predictivo que permita predecir el precio del inmueble en función de dichos parámetros. Las **variables** seleccionadas son adecuadas para este modelo, exceptuando un bajo nivel de significancia en el número de camas y la política de cancelación, por lo que se podría excluir del modelo. Este modelo predice bien el precio según las métricas seleccionadas. El modelo en ciertos barrios podría no predecir de forma realista, por lo que se podría mejorar a posteriori. Sin embargo, la mayoría de las métricas seleccionadas fueron adecuadas. En conclusión, este estudio permitiría de manera general a los propietarios seleccionar el precio de los inmuebles en función de las características mencionadas anteriormente.

En definitiva, el desarrollo de este proyecto ha permitido poner en práctica en un entorno próximo a la realidad las herramientas y recursos aprendidos durante el bootcamp de KeepCoding. Además, ha fomentado el trabajo en equipo y la participación entre los diferentes miembros del proyecto, de manera que se ha promovido la colaboración así como la persecución de objetivos comunes.