



# Reporte de Big Data.

Análisis de Airbnb Data: Madrid

Febrero del 2023

---

## Team Seaborn Girls

Cherry Reynoso

Cintia Guerrero

Núria Orgaz

Paola Villalba

## Visión general

Airbnb es una plataforma de uso global que nos permite buscar alojamiento tanto a turistas como a particulares principalmente en las ciudades más importantes de cada país, con variedad de precios y de tipos de alojamientos.

En este reporte usaremos datos de Airbnb Madrid por medio de una dataset (alojado [aquí](#)) y en el que pretendemos encontrar relación entre el precio y el tipo de alojamiento, localización y valoraciones de los usuarios y con ello determinar qué atributos pueden incrementar el precio.

## Objetivos

1. Comprender los datos y la información contenida para poder hacer una valoración importante de la información que tenemos y su correlación a través de la exploración, el análisis y la visualización de métricas.
2. Obtener los KPIs relevantes y encontrar un algoritmo que prediga el precio de un inmueble según los datos obtenidos.
3. Comprobar las hipótesis planteadas en el inicio del proyecto, según las variables disponibles:

H1 → Los precios de los alojamientos situados en el Centro (barrio) són mayores que los del resto de barrios.

H0 → Los precios de los alojamientos del centro no són más caros que los alojamientos de otros barrios

H1 → Los precios de los alojamientos que pertenecen a un host que lleva más tiempo serán más altos

H0 → Los precios de los alojamientos que pertenecen a un host que lleva más tiempo no serán más altos → Los alojamientos que pertenecen a un host que lleva más tiempo són más baratos

H1 → El precio de los alojamientos aumentará cuando el número de huéspedes/ número de reviews / puntuación de las reviews / ratio de respuesta del host / habitaciones / baños / camas aumente

H0 → El precio de los alojamientos no aumentará cuando el número de huéspedes/ reviews / puntuación de las review/ ratio de respuesta del host / baños / camas aumente

H1 → El precio de las casas és mayor que el del resto de alojamientos

H0 → El precio de las casas no és mayor que el del resto de alojamientos

H1 → El precio de los alojamientos que se alquilan enteros es superior al precio de habitaciones compartidas o habitaciones privadas.

H0 → El precio de los alojamientos que se alquilan enteros no es superior al precio de habitaciones compartidas o habitaciones privadas.

H1 → El precio de los alojamientos que tienen una cama real es superior al precio del resto de tipos de camas.

H0 → El precio de los alojamientos que tienen una cama real no es superior al precio del resto de tipos de camas.

H1 → El precio de los alojamientos que piden fianza / tasa de limpieza tienen un precio superior a los alojamientos que no la piden.

H0 → El precio de los alojamientos que piden fianza / tasa de limpieza no tienen un precio superior a los alojamientos que no la piden.

H1 → El precio de los alojamientos que pertenecen a host con más alojamientos alquilados es superior al de un alojamiento que pertenece a un host que no tiene más alojamientos alquilados.

H0 → El precio de los alojamientos que pertenecen a host con más alojamientos alquilados no es superior al de un alojamiento que pertenece a un host que no tiene más alojamientos alquilados.

## Metodología

Dentro del equipo y por la naturaleza del proyecto, hemos usado la **metodología Kanban**, dividiendo las tareas y colocándolas en un tablero en tres categorías: "To Do", "Doing" y "Done".



1.1 Tablero del equipo alojado en Trello donde colocamos flujo de las actividades pendientes y hechas.

## Análisis.

Nuestro dataset de Airbnb está conformado por:

- 14780 entradas, algunas con valores nulos
- 89 columnas. Es necesario reducir el número de variables.
- En la variable de 'City', se encuentran estos países: 'United Kingdom','Spain','United States', 'Canada', 'Switzerland', 'Hong Kong', 'Cuba', 'Germany', 'Belgium', 'Netherlands', 'Italy', 'Australia', 'Austria', 'France', 'Denmark', 'Ireland', 'Greece'.

Sólo queremos aquellas que sean de España y en concreto de la comunidad de Madrid.

Con este primer acercamiento que hemos hecho en el equipo, sabemos que el primer punto será la limpieza de nuestra dataset tanto de valores nulos, variables innecesaria, revisión del tipo de muestra variables y creación de variables calculadas

## Hitos

### I. Arquitectura y validación de los datos.

En esta primera parte , lo que hicimos fue dar una primera depuración del dataset, analizando las columnas que serán necesarias para su posterior análisis en el resto de los hitos y cuáles no, por ello concluimos que las columnas con las que nos quedamos serán 23 del total del dataset, donde se incluyen las columnas de latitud y longitud para el mapa en Tableau , que es la herramienta que hemos decidido utilizar para la visualización del proyecto.

En la depuración inicial se determinó que las columnas que contengan muchos datos NA o campos vacíos serían descartados por dar poca información, además las columnas de texto también fueron eliminadas.

Con este primer filtro, pasamos el csv a la siguiente fase del Análisis Exploratorio para un análisis más profundo.

## II. Análisis Exploratorio

Para la depuración de la matriz y el análisis exploratorio de las variables se ha utilizado la herramienta R studio.

El primer paso es **importar el csv** con las columnas seleccionadas y comprobar que se ha cargado correctamente revisando la dimensión de la matriz. Creamos una nueva variable fuera de la matriz llamada `nombre_columnas` para **comprobar el nombre de las columnas importadas**.

```
1 datos <- read.csv("C:/Users/cinti/OneDrive/Escritorio/Proyecto/airbnb_def.v0.csv", sep= ";")
2
3
4 #Comprovamos que se haya cargado correctamente
5 dim(datos)
6 nombre_columnas <- colnames(datos)
7 print(nombre_columnas)
8
```

A partir del nombre de las columnas importadas, se revisa si el nombre es correcto y se identifica con los datos que incluye o no. En este caso se ha realizado un cambio en el nombre de la primera columna correspondiente a la variable ID. Se ha cambiado el nombre importado de "i..ID" por "ID" para facilitar la redacción del script. El resto de columnas tienen un nombre claro e identificativo.

```
9 #Renombramos los nombres de las columnas que queramos i comprobamos:
10
11 colnames(datos)[colnames(datos)=="i..ID"] <- "ID"
12 nombre_columnas <- colnames(datos)
13 print(nombre_columnas)
14
```

Seguimos el análisis con la **revisión del tipo de variable** que se identifica con cada columna, la revisión de los datos que incluye cada variable, para ver si se ajusta la tipo de variable, y se modifica el tipo de variable si no es correcto. Después del cambio del tipo de variable se vuelven a revisar los datos que incluye la variable para verificar que no han habido problemas y que los datos se mantienen. También se comprueba si el tipo de variable se ha modificado correctamente volviendo a ejecutar la función "class".

```

25
26 class(datos$Host.Since)
27 datos$Host.Since
28 datos$Host.Since <- as.Date(datos$Host.Since, format = "%d/%m/%y")
29 datos$Host.Since
30 class(datos$Host.Since)
31
32 class(datos$Host.Response.Rate)
33 datos$Host.Response.Rate
34 datos$Host.Response.Rate <- as.numeric(datos$Host.Response.Rate)
35 datos$Host.Response.Rate
36 class(datos$Host.Response.Rate)
37
38 class(datos$Neighbourhood.Cleansed)
39 datos$Neighbourhood.Cleansed
40 datos$Neighbourhood.Cleansed <- as.factor(datos$Neighbourhood.Cleansed)
41 datos$Neighbourhood.Cleansed
42 class(datos$Neighbourhood.Cleansed)
43

```

En este punto la mayoría de variables se recodifican a un nuevo tipo, especialmente a tipo numérica, factor y de fecha. Las variables Zipcode, Latitude y Longitude se mantienen en tipo character porque el formato de datos introducidos no es numérico pero tampoco es texto y para el análisis inicial no se tienen en cuenta, solo se usan para la representación con Tableau.

Una vez asignado el tipo de variable a cada columna se procede al **análisis individual de cada variable**. La primera variable a analizar es la columna ID que corresponde al código asignado a cada alojamiento. Ésta variable se ha mantenido como **variable de control** y se analiza que no haya valores que faltan y que no contenga ningún duplicado.

```

157 sum(is.na(datos$ID)) #0 Na
158
159 #en este caso analizamos si hay valores repetidos, ya que la variable ID
160 #es una variable de control
161 any(duplicated(datos$ID)) #No hay valores repetidos = no se incluye el mismo inmueble 2 veces
162

```

Para las **columnas numéricas** se comprueba el número de valores faltantes y analiza el recuento de respuestas, la suma, la media, los cuartiles, la desviación estándar y los valores mínimos y los máximos. Seguidamente se ordena la variable y se realiza el tratamiento de los valores perdidos para eliminarlos y se vuelve a hacer el análisis de los principales estimadores indicados anteriormente y un gráfico si se considera necesario para acabar de ver la representación de la variable.

El tratamiento de los valores que faltan se realiza según la necesidad de cada variable, en la mayoría de casos se ha imputado la mediana, aunque en algunos casos se ha imputado la media, según la coherencia de los datos.

```

220
221 summary(is.na(datos$Bedrooms)) #25 NA y 14755 complete
222 summary(datos$Bedrooms)
223 sd(datos$Bedrooms, na.rm = TRUE) #0,9
224
225 #Ordenamos la variable
226 datos$Bedrooms[!is.na(datos$Bedrooms)] <- sort(datos$Bedrooms[!is.na(datos$Bedrooms)])
227
228 #Creamos una nueva variable sin NA, imputando datos de la mediana
229 median_Bedrooms <- median(datos$Bedrooms, na.rm = TRUE)
230 datos$Bedrooms_sinNA <- ifelse(is.na(datos$Bedrooms), median_Bedrooms, datos$Bedrooms)
231

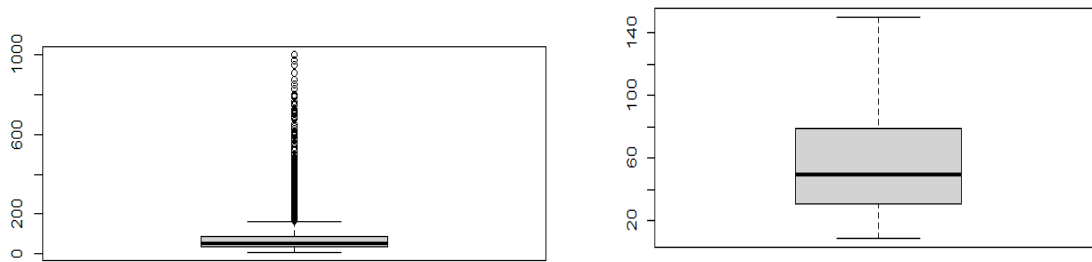
```

Donde sí se han eliminado las filas correspondientes a los datos faltantes es en la variable dependiente del estudio "Price", así como se han eliminado las filas que representaban outliers: precios superiores a 150€/noche. En este punto se pasa de una matriz de 14.780 filas a una matriz de 13.683 filas, reduciendo la desviación estándar de 72 a 32,5.

```

255
256 #Analizamos el numero de respuestas, de NA y los principales indicadores
257 summary(is.na(datos$Price)) #17 NA y 1463 complete
258 summary(datos$Price)
259 sd(datos$Price, na.rm = TRUE) #desviación de 72 (muy elevada)
260
261 #Analizamos los outliers
262 boxplot(datos$Price) #Outliers a partir de 150€/noche
263
264 #Creamos una nueva variable que contenga solo los datos =< a 150
265 #y analizamos el numero de respuestas que nos quedaria
266 #también revisamos de nuevo el boxplot y la desviación estandar
267 Price_noOutliers <- datos$Price[datos$Price<= 150]
268 summary(is.na(Price_noOutliers)) #Quedan 13683 casos de 14780 (aceptable)
269 boxplot(Price_noOutliers)
270 sd(Price_noOutliers, na.rm = TRUE) #desviación se ha reducido a 32,5
271
272 #Ordenamos la variable
273 datos$Price[!is.na(datos$Price)] <- sort(datos$Price[!is.na(datos$Price)])
274
275 #Pasamos los valores >150 a NA y despues los eliminamos
276 datos$Price_sinNA <- ifelse(datos[, "Price"] > 150, NA, datos[, "Price"])
277 summary(is.na(datos$Price_sinNA))
278 datos <- datos[!is.na(datos[, "Price_sinNA"]), ]
279 dim(datos)
280

```



De las variables Cleaning fee y Security deposit, además, se crean dos columnas booleanas, una para cada variable, según si el alojamiento pide ese pago extra (1) o no (0).

```

330
331 #De las variables cleaning fee y Security Deposit creamos dos variables booleanas
332 #Que contengan 0 y 1 segun si tienen o no esas fianzas
333
334 datos$Security.Deposit_bool <- ifelse(datos$Security.Deposit == 0,0,1)
335 datos$Cleaning.fee_bool <- ifelse(datos$Cleaning.Fee == 0,0,1)
336

```

Además, en el caso de la variable Bathroom que en la matriz incluyen decimales separados por ".", se ha generado una nueva variable character separando los decimales por comas, por si durante otras fases del proceso de análisis fuera necesario tenerla en este formato.

```

216
217 datos$Bathrooms_decimal <- gsub("\\.", ",", datos$Bathrooms)
218 class(datos$Bathrooms_decimal)
219

```

De las tres **variables de tipo character**, la única que contiene valores faltantes es la variable zipcode, pero no se eliminan porque es una variable que solo se usa en la representación y eliminarlos solo haría perder casos (412) que sí aportan toda la demás información. Tampoco se recodifican por la dificultad de asignar un código postal según la ubicación.

En el caso de la **variable date** "Host.Since" se detectan 2 missing values, y se eliminan porque imputarlas no tiene mucho sentido ya que desconocemos la fecha exacta y la pérdida es muy leve, por lo que la matriz se queda con **13.681 filas**. De esta variable se genera otra columna llamada



ano\_Host\_Since, de tipo numérica, que contiene solo el año de registro del Host.

Para las **variables factor** se realizará una tabla de contingencias y el comando “levels” para analizar cuántas categorías de la variable hay, como están redactadas y cuáles de ellas contienen datos. También comprobamos si la variable contiene valores faltantes. En el caso de las variables factor no se encuentran, pero sí categorías de variable que contienen 0 registros y que se eliminan. Seguidamente, se normalizan los valores, eliminando tildes y corrigiendo los nombres escritos de forma errónea.

```

445 table(datos$Neighbourhood.Cleansed)
446 levels(datos$Neighbourhood.Cleansed)
447 sum(is.na(datos$Neighbourhood.Cleansed))
448
449 #eliminamos las categorías con 0 registros
450 datos$Neighbourhood.Cleansed <- droplevels(datos$Neighbourhood.Cleansed)
451 levels(datos$Neighbourhood.Cleansed)
452
453 #Normalizamos los valores
454 datos$Neighbourhood.Cleansed <- as.factor(recode(toupper(datos$Neighbourhood.Cleansed),
455                                     "ARGÁELLES" = 'ARGUELLES',
456                                     "CÁRMENES" = 'CARMENES',
457                                     "CASCO HISTÁRICO DE BARAJAS" = 'CASCO HISTORICO DE BARAJAS',
458                                     "CASCO HISTÁRICO DE VALLECAS" = 'CASCO HISTORICO DE VALLECAS',
459                                     "CASCO HISTÁRICO DE VICÁLVARO" = 'CASCO HISTORICO DE VICALVARO',
460                                     "CIUDAD JARDÁN" = 'CIUDAD JARDIN',
461                                     "CONCEPCIÁN" = 'CONCEPCION',
462                                     "EL PLANTÁN" = 'EL PLANTIO',

```

Una vez se tiene la variable normalizada y sin categorías vacías se ordena y se genera una nueva columna numérica, asignando un valor numérico a cada una de las categorías de texto de la inicial.

```

484 #ordenamos la variable por orden alfabetico --> despues de pasarla a numerica
485 datos$Neighbourhood.Cleansed <- reorder(datos$Neighbourhood.Cleansed, datos$Neighbourhood.Cleansed, FUN=sort)
486
487 #Pasar la variable a numerica
488 datos$Neighbourhood.Cleansed_num <- as.numeric(recode(datos$Neighbourhood.Cleansed,
489                                     "ABRANTES" = 1,
490                                     "ACACIAS" = 2,
491                                     "ADELFAS" = 3,
492                                     "AEROPUERTO" = 4,
493                                     "AGUILAS" = 5,
494                                     "ALAMEDA DE OSUNA" = 6,
495                                     "ALMAGRO" = 7,
496                                     "ALMENARA" = 8,
497                                     "ALMENDRALES" = 9,
498                                     "ALUCHE" = 10,
499                                     "AMBROZ" = 11,
500                                     "AMPOSTA" = 12,
501                                     "APOSTOL SANTIAGO" = 13,

```

Éste proceso se ha replicado en todas las variables factor excepto en las variables Country y City, en las que se eliminaron las filas que no correspondían a Spain y Madrid respectivamente, y no se crearon variables numéricas porque solo se quedaban con una única categoría. La matriz final contiene un total de 12.375 filas.

```

432 table(datos$City)
433 levels(datos$City)
434 sum(is.na(datos$City)) # 0 NA
435
436 #Normalizamos los valores correspondientes a Madrid
437 datos$City <- as.factor(recode(toupper(datos$City), Madrid = "Madrid", Madri = "Madrid", madrid = "Madrid", MADRID = "Madrid"))
438 datos$City <- as.factor(recode(toupper(datos$City), MADRID = "Madrid", MADRI = "Madrid"))
439 datos$City <- as.factor(sub("^(^,)+.*", "\\1", datos$City))
440
441 #Eliminamos las filas que pertenecen a otra ciudad y no nos interesan
442 datos <- datos[datos$City %in% "MADRID", ]
443 datos$City <- droplevels(datos$City)
444

```

Para la variable Property.Type se genera una nueva variable Property.Type\_Rec agrupando las siguientes variables que contienen valores < 10 en diferentes categorías, quedando la nueva variable con 10 categorías, de las 22 que tiene la variable original. A partir de esta nueva variable se genera otra variable numérica para analizar la correlación.

```

660 table(datos$Property.Type)
661 levels(datos$Property.Type)
662 sum(is.na(datos$Property.Type))
663 datos$Property.Type <- as.factor(toupper(datos$Property.Type))
664
665 #Agrupar las categorías con poca representación
666 datos$Property.Type_Rec <- recode(datos$Property.Type,
667                                "CASA PARTICULAR" = "HOUSE",
668                                "VILLA" = "HOUSE",
669                                "SERVICED APARTMENT" = "APARTMENT",
670                                "BOAT" = "OTHER",
671                                "BOUTIQUE HOTEL" = "OTHER",
672                                "BUNGALOW" = "OTHER",
673                                "CAMPER/RV" = "OTHER",
674                                "EARTH HOUSE" = "OTHER",
675                                "GUEST SUITE" = "OTHER",
676                                "TENT" = "OTHER",
677                                "TIMESHARE" = "OTHER",
678                                "TOWNHOUSE" = "OTHER")
679

```

Para la variable Bed.Type también se agrupan las categorías con una representación baja (< 33) generando una nueva variable con dos categorías y a partir de la nueva variable se genera otra nueva variable numérica para hacer la correlación.

```

711 table(datos$Bed.Type)
712 levels(datos$Bed.Type)
713 sum(is.na(datos$Bed.Type))
714 datos$Bed.Type <- as.factor(toupper(datos$Bed.Type))
715
716 #Agrupamos las categorias
717 datos$Bed.Type_REC <- as.factor(recode(toupper(datos$Bed.Type),
718                                     "AIRBED" = 'REAL BED',
719                                     "COUCH" = 'SOFA OR SIMILAR',
720                                     "FUTON" = 'SOFA OR SIMILAR',
721                                     "PULL-OUT SOFA" = 'SOFA OR SIMILAR'))
722 levels(datos$Bed.Type_REC)
723
724 #Pasar a numerica
725 datos$Bed.Type_ReNum <- as.numeric(recode(datos$Bed.Type_REC, "REAL BED" = 1,
726                                     "SOFA OR SIMILAR" = 2))
727
728 table(datos$Bed.Type_ReNum)
729

```

Posteriormente se exporta la matriz para poder trabajar con ella y se realiza la matriz de correlaciones para analizar las correlaciones entre las distintas variables y poder afirmar o desmentir las hipótesis iniciales.

```

731 #Exportamos la matriz en formato csv
732 write.csv(datos, file = "airbnb_def.csv")
733
734 #Hacemos la matriz de correlaciones para ver las correlaciones entre las variables
735 datos[, -which(names(datos) == "Price")] <- lapply(datos[, -which(names(datos) == "Price")], as.numeric)
736 cor(datos[, -which(names(datos) == "Price")], datos$Price)
737

```

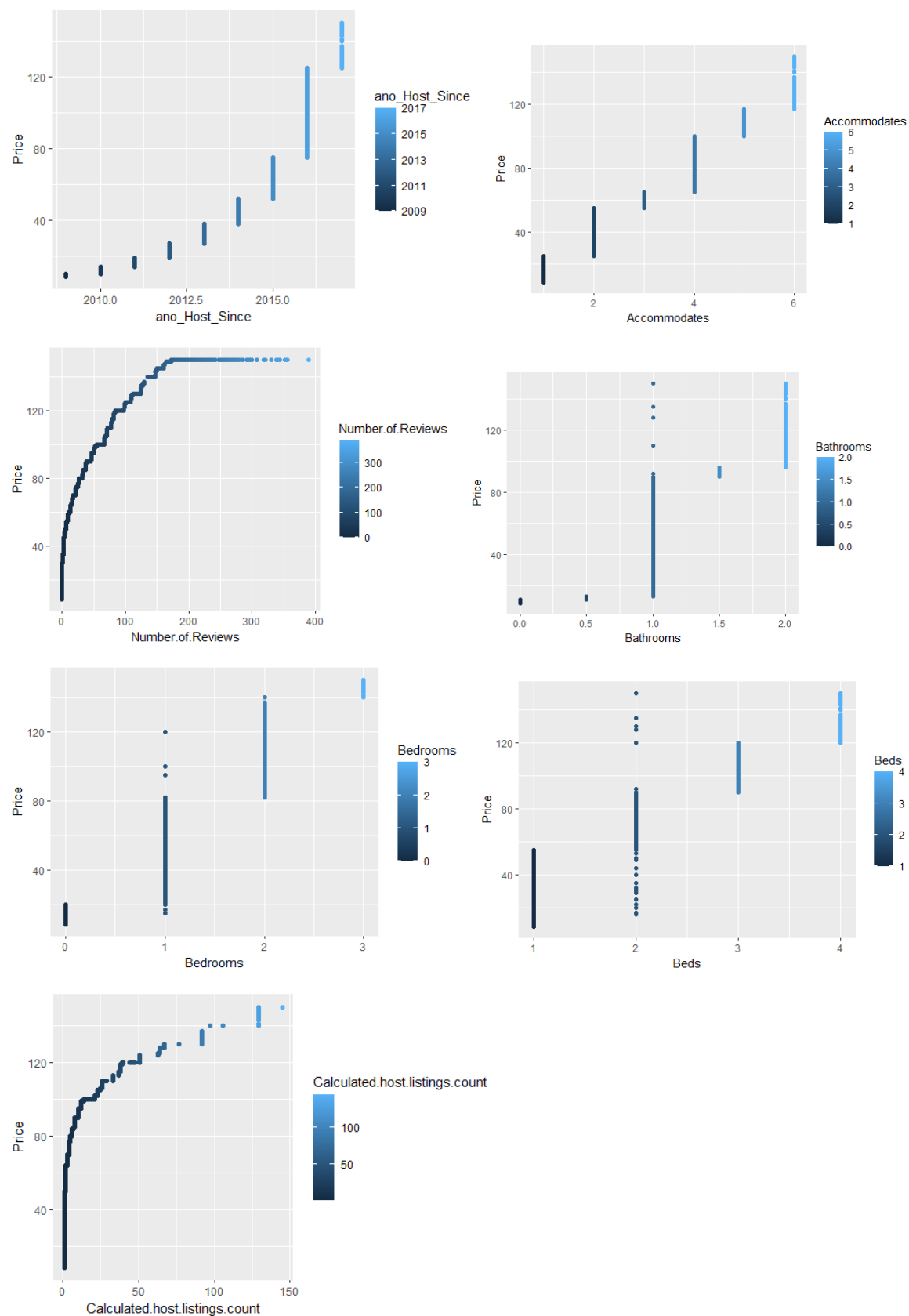
Los resultados obtenidos nos indican la relación entre variables.

	[,1]
ID	-0.001496909
Host.Since	0.926383954
Neighbourhood.Cleansed	0.008237984
Neighbourhood.Group.Cleansed	-0.010857193
City	NA
Zipcode	NA
Country	NA
Latitude	NA
Longitude	NA
Property.Type	0.009365990
Room.Type	0.007478357
Accommodates	0.964533468
Bed.Type	-0.004877170
Security.Deposit	0.918720999
Cleaning.Fee	0.860686139
Number.of.Reviews	0.874768025
Review.Scores.Rating	0.137347378
Host.Response.Rate	0.460859201
Bathrooms	0.793464362
Bathrooms_decimal	NA
Bedrooms	0.846353962
Beds	0.939611405
Security.Deposit_bool	0.818532037
Cleaning.fee_bool	0.757759425
Calculated.host.listings.count	0.731113026
ano_Host_Since	0.917948631
Neighbourhood.Cleansed_num	0.008237984
Neighbourhood.Group.Cleansed_num	-0.010857193
Property.Type_Rec	0.007193320
Property.Type_RecNum	0.013731786
Room.Type_Num	0.007478357
Bed.Type_REC	0.010982025
Bed.Type_ReCNum	0.010982025

Las variables con más correlación son:

- Host Since
- Accomodates
- Number of reviews
- Bathrooms
- Bedrooms
- Beds
- Calculated.host.listings.count

Por último, se realizan los gráficos correspondientes para visualizar la relación entre la variable Precio y las variables con más correlación que influyen:



### III. Visualización de las métricas

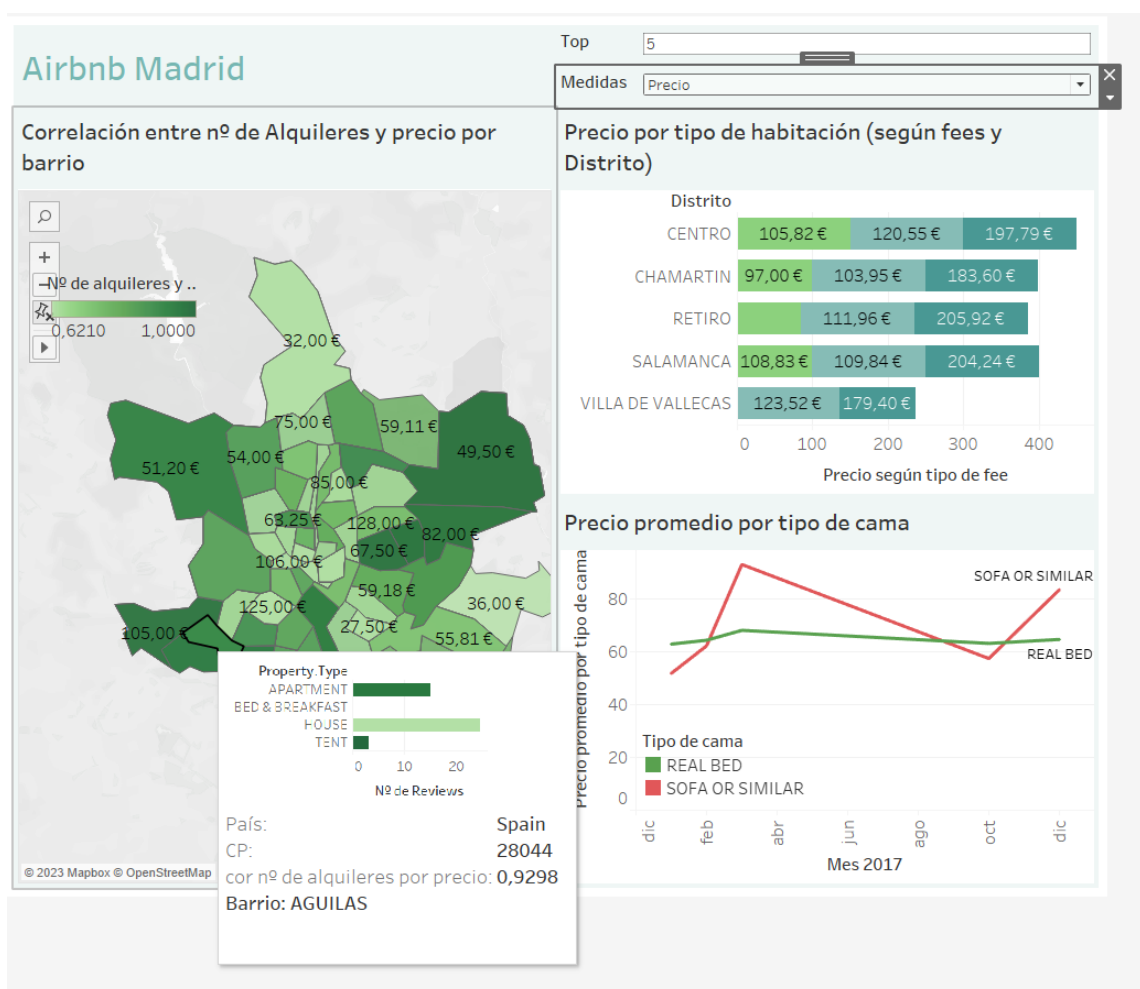
Después del análisis exploratorio, se pudo determinar que las variables más con más correlación (y que se mencionan en el apartado anterior):

Host Since, Accomodates, Number of Reviews, bathrooms, bedrooms, beds y Calculated Host Listing account.

Para la visualización de nuestras métricas usamos la aplicación Tableau Desktop versión 2022.3.1.

Apegandonos a un estilo sencillo y comprensible, hicimos un dashboard conformado por tres hojas:

- Mapa de correlación entre alquileres por host y el precio por barrio.
- Gráfica de barras del precio por tipo de habitación según los fees y los Distritos Top.
- Gráfica de líneas discretas, en relación del precio por tipo de cama a lo largo del año 2017.



### 1) Mapa de *Correlación entre nº de Alquileres y precio por barrio*

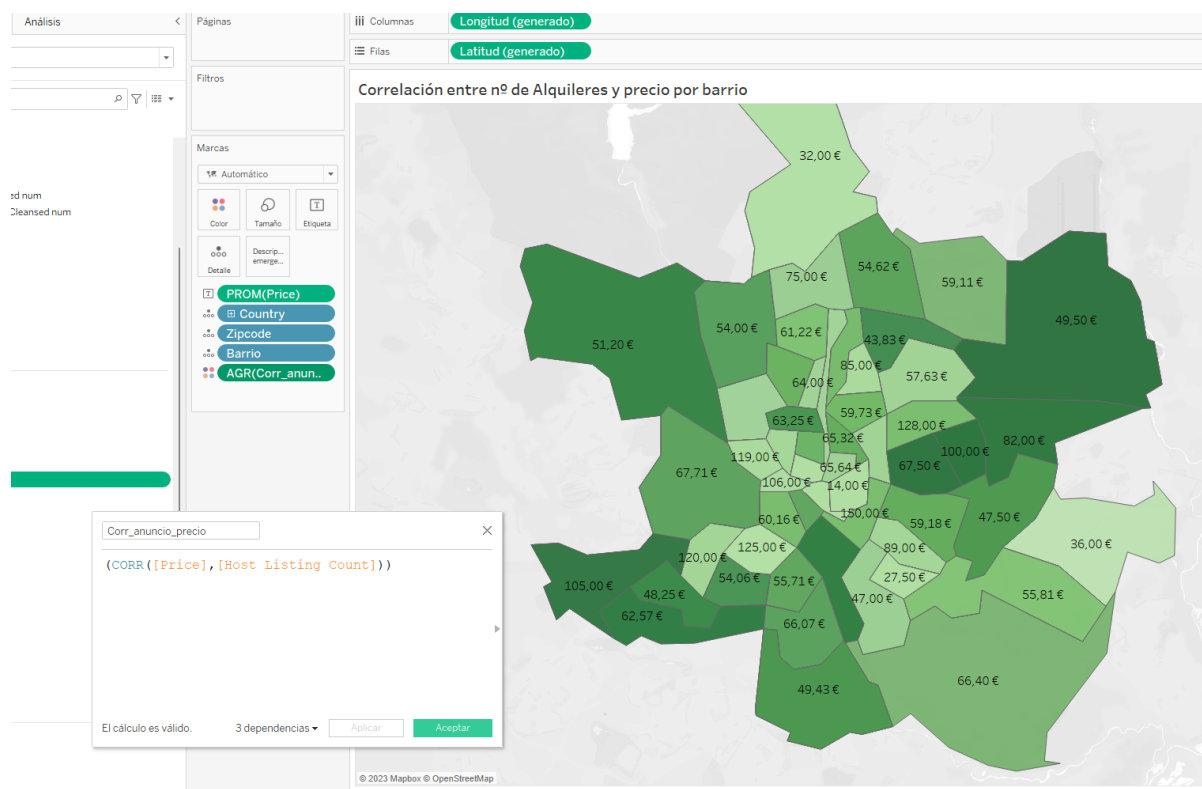
Para la creación de este mapa utilizamos las variables :

- Precio (promedio)
- Calculated Host Listing count (dentro de un campo calculado:  $(\text{CORR}([\text{Price}], [\text{Host Listing Count}])))$
- Zipcode, Neighbourhood Cleansed (para generar el mapa)

La variable "Calculated Host Listing account" se refiere al número de propiedades en alquiler que tiene cada Host. A mayor número de propiedades de un host, el rating del host aumenta así como su probabilidades de convertirse en "Super Host" por lo que es un factor relacionado con el precio. (El host conoce mejor el mercado).

Para reflejar esto, por medio de un campo calculado, se hizo una correlación entre la variable 'Calculated Host Listing account' con la variable del 'Price' y poder reflejarla dentro de un mapa por barrio.

- (CORR([Price],[Host Listing Count]))



Al mismo tiempo, entre más anuncios de propiedad tiene un host, el número de reviews aumenta. Creamos una gráfica auxiliar de barras.

#### a. Gráfica auxiliar de Tipo de alojamiento por reviews y respuesta del host

En esta gráfica auxiliar utilizamos las variables:

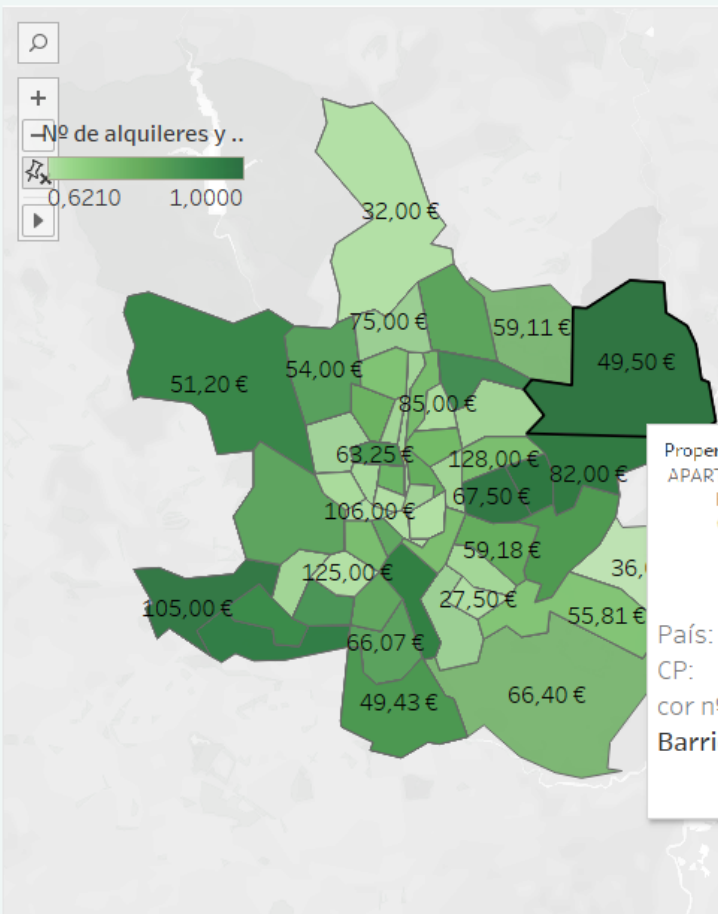
- Property Type. (tipo de alojamiento)
- Número de Reviews
- Host Response Rate(promedio)

Gracias a esta gráfica auxiliar podemos comprobar que aquellos barrios con una mayor correlación entre Host listings y precio, también estaba ligado a un mayor número de reviews según el tipo de propiedad alquilada así como el promedio del rating score del host.



## Airbnb Madrid

Correlación entre nº de Alquileres y precio por barrio



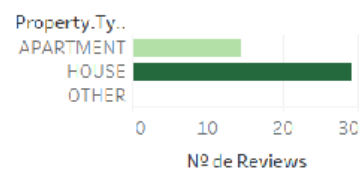
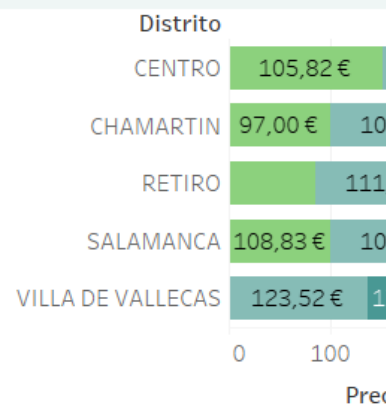
Top

5

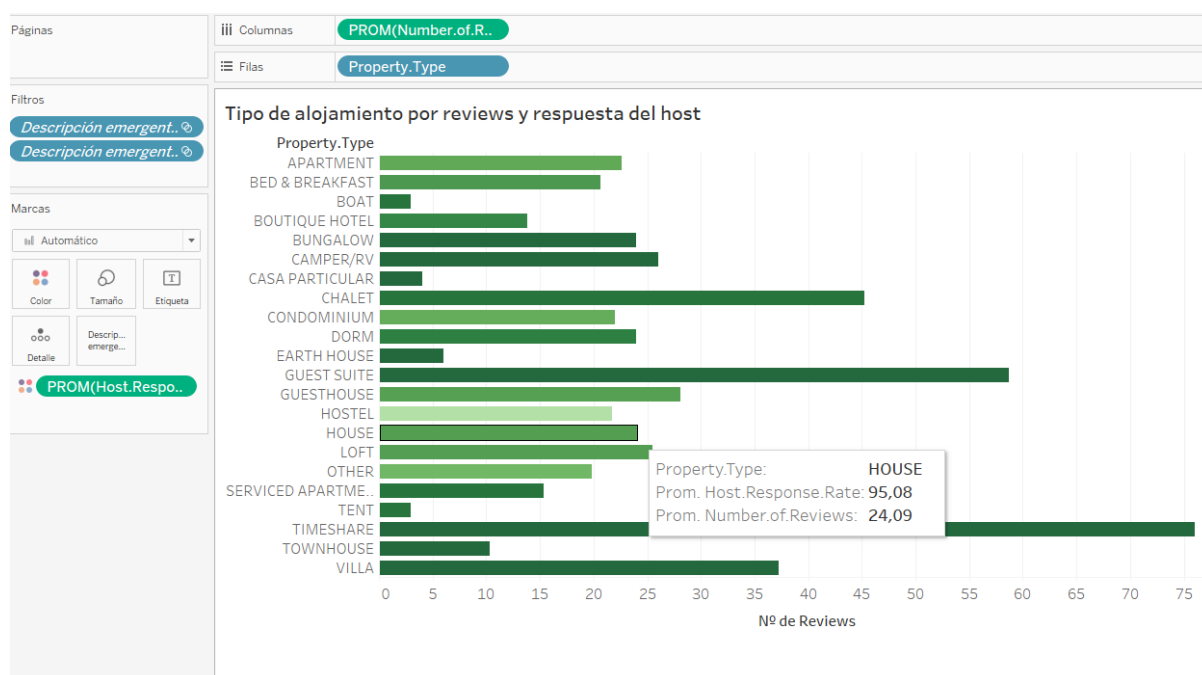
Medidas

Precio

Precio por tipo de habitación (Distrito)



País: Spain  
 CP: 28042  
 cor nº de alquileres por precio: 0,9976  
 Barrio: AEROPUERTO



## 2) Gráfico de barras. Precio por tipo de habitación (según fees y Distrito)

Con las siguientes variables, se creó una gráfica de barras con vistas interactiva.

Variables <ul style="list-style-type: none"> <li>- Distrito.</li> <li>- Room. Type</li> <li>- Price ( Promedio)</li> <li>- Cleaning Fee</li> <li>- Security Deposit</li> <li>- Accomodates</li> </ul>	Parametros: <ul style="list-style-type: none"> <li>- Top N</li> <li>- Medidas (Valores de Precio, limpieza, y Depósito)</li> </ul>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------



A través de esta gráfica, se pudo observar Top de los distritos con los precios totales más altos según el tipo de fee (Total, limpieza, deposito)

Se crearon varios campos calculados.

El primero para calcular el precio total fue:

$AVG([Price]) + AVG([Cleaning.Fee]) + AVG([Security.Deposit])$

El segundo un campo calculado que nos permitiera escoger el tipo de fee:

CASE [Parámetros].[Medidas]

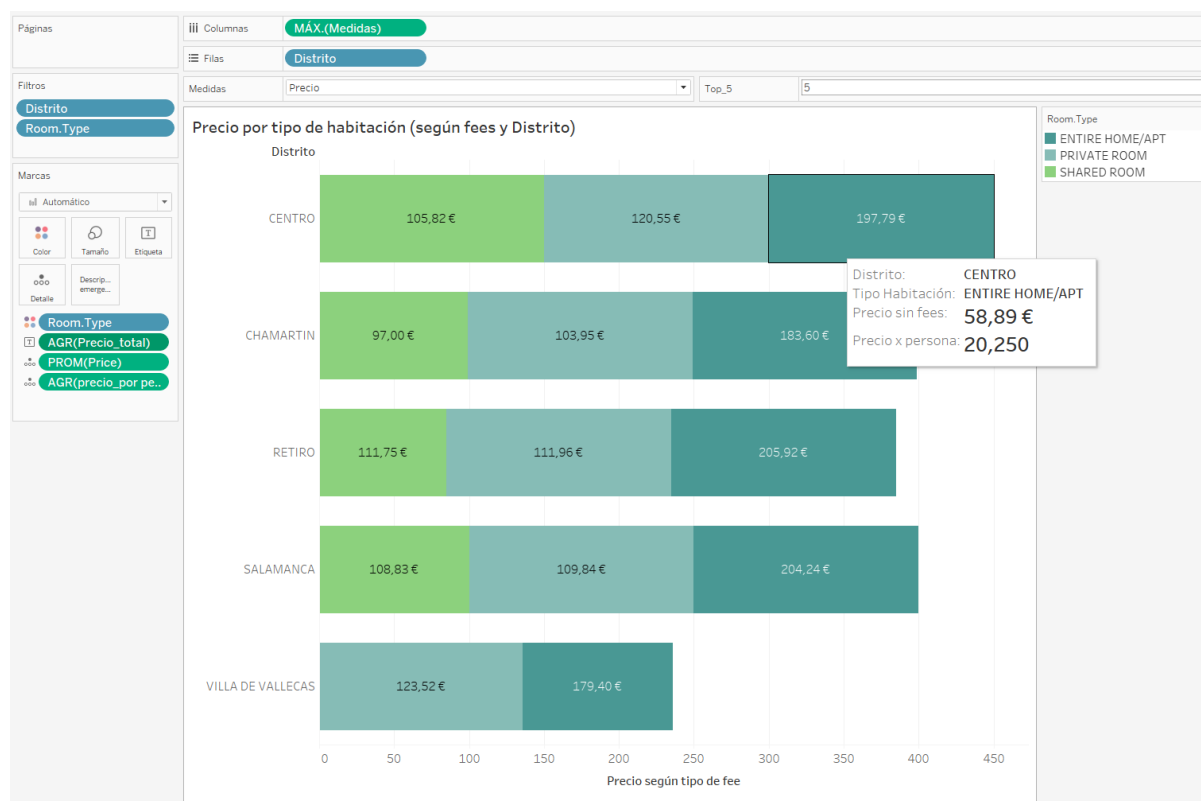
WHEN 0 THEN [Price]

WHEN 1 THEN [Cleaning.Fee]

WHEN 2 THEN [Security.Deposit]

END

La gráfica permite observar que los distritos con los precios más altos son: Centro, Chamartin, Retiro, Salamanca y Villa de Vallecas.



Sin embargo, era necesario completar la información para saber el precio sin costes extras y además, el precio por persona.

Para calcular el precio por persona, se utilizó el siguiente calculo:

$$\text{AVG}([\text{Price}]) / \text{AVG}([\text{Accommodates}])$$

Estos dos elementos salen en la descripción emergente completando la diferencia de precios que puede haber según que gastos extras se incluyan y cuanto es el coste por persona.

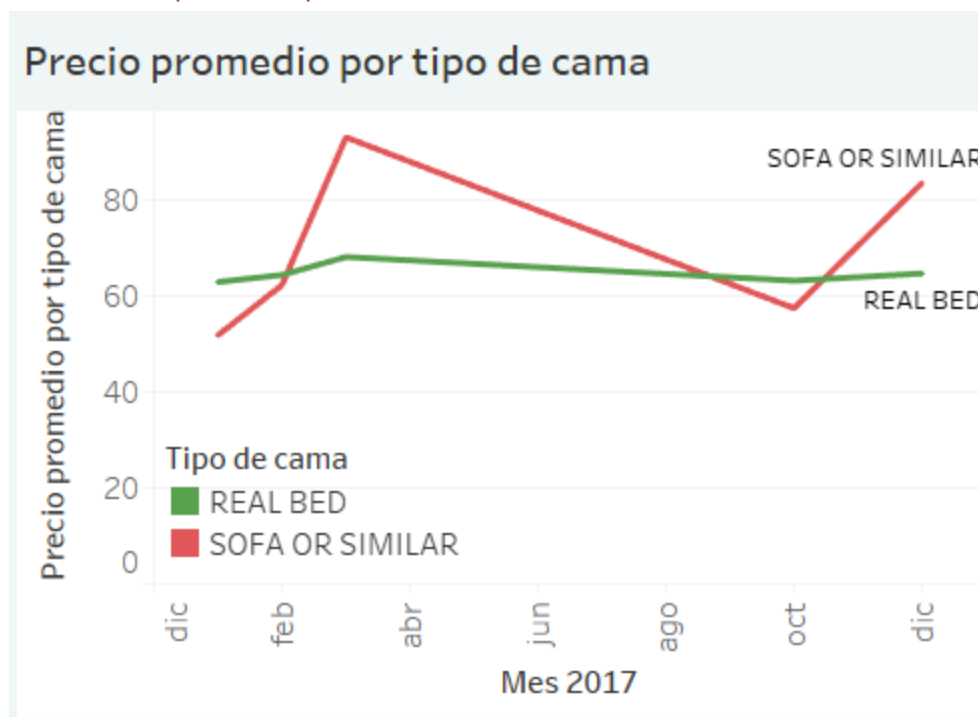
### 3) Gráfico lineal discreto. Precio promedio por tipo de cama

Por medio de una gráfica lineal se reflejó la relación de las variables:

- Bed.Type
- Host Since
- Price (promedio)

La variable Host Since es una variable que se refiere al año en que un host registro una propiedad. Los datos más abundantes están en el año 2017.

Con la variable tipo de cama (Real o Sofa y similares) y el precio, se pudo observar como durante los meses del 2017 hubo momentos en que el precio de "Sofa y similares" subió a diferencia de la entrada "Cama Real" y que esta relación dependía de los meses (Marzo-abril, Diciembre) por lo que podíamos deducir un aumento de demanda de todo tipo de alquileres durante esas fechas.



#### IV. Pre-Procesamiento y Modelado

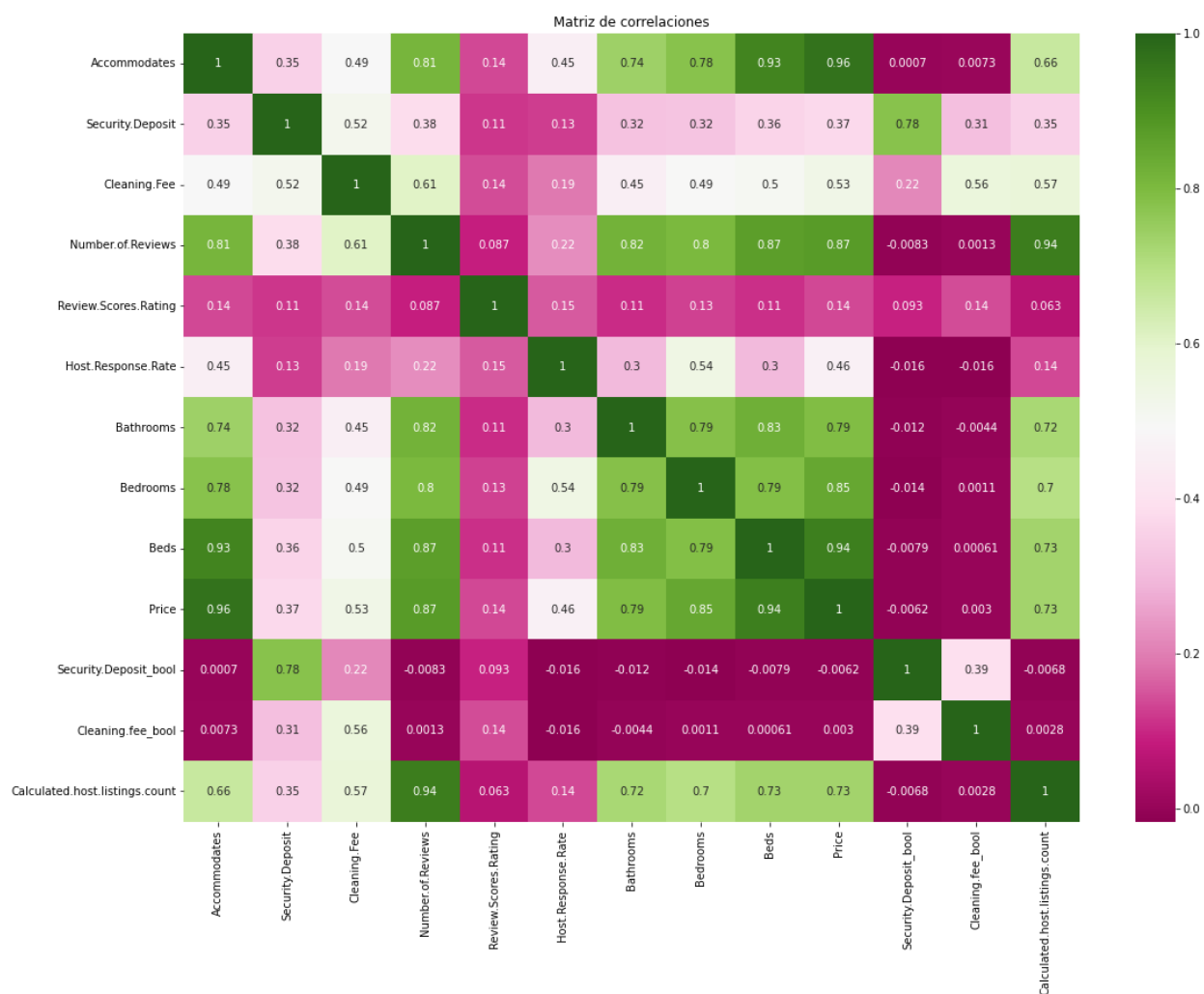
El objetivo es crear un **modelo de regresión** lineal que pueda predecir el precio de un alojamiento anunciado en la página de Airbnb. Se utilizarán python y varias librerías de este lenguaje. En este apartado se utilizará el dataset limpiado en los anteriores apartados. A parte, se explorará la utilidad de las variables disponibles para este modelo.

##### Elección de las variables

El dataset dispone de **25 variables**, de las cuales diez son variables cualitativas y quince son variables numéricas. Algunas de estas variables se pueden eliminar ya

que no van a aportar información útil para el modelo. No se incluirán éstas por las siguientes razones:

- **ID:** es el identificador de la fila, no da información al algoritmo.
- **Zipcode:** el código postal, que es de tipo numérico pero no se puede tomar como número. El algoritmo interpretaría que hay un orden o un valor cuando no lo hay. Además, para usarlo como una variable categórica ya disponemos de otras variables que nos dan la misma información.
- **Latitude y Longitude:** por la misma razón que con el código postal. Ya se dispone de los barrios, y además el algoritmo no es capaz de interpretar que estas dos variables van en conjunto, pues aquí están separadas en dos columnas diferentes.
- **City:** solo hay un valor posible de ciudad en este dataset, Madrid
- **Country:** solo hay un valor posible de país en este dataset, España



Tras eliminar esas columnas, se calculan las correlaciones entre las variables numéricas y se hace la matriz de correlaciones. Se puede ver que la mayoría están correlacionadas con el precio, excepto `Security.Deposit_bool` y `Cleaning.fee_bool`, que son columnas calculadas en base a si el listing tiene depósito de seguridad o gastos de limpieza respectivamente. Siendo valores tan cercanos a 0, se pueden eliminar del modelo.

En cuanto a las demás variables, es llamativo ver la **alta correlación** entre las variables. Se cumplen las hipótesis planteadas; el precio es mayor cuantas más personas acomoda, por lo tanto también afectan el número de camas, habitaciones y baños. Es coherente, puesto que son inmuebles más grandes. También es interesante ver que a mayor número de propiedades tiene el anfitrión anunciándose en Airbnb, mayor será el precio de un inmueble suyo.

### Pre-procesamiento

Las variables categóricas restantes han de ser **transformadas en dummies** para poder utilizarlas en el algoritmo. Los dummies convierten una variable categórica en series de variables binarias para facilitar su cuantificación. Significa también que si una columna contiene 25 valores diferentes, se tendrán que crear 25 columnas nuevas. Después de transformar las variables categóricas, el dataframe contiene 63 columnas.

Antes de seguir con el modelo, hay que dividir el dataset en dos con una proporción desigual, del 70%-30%. Una parte con la que se entrenará el modelo, **train**, y otra con datos que no ha visto aún, **test**, para comprobar que predice bien y que **no hay overfitting**. El overfitting ocurre cuando el modelo se ajusta muy bien a los datos con los que ha sido entrenado, pero que se desempeña mal con datos nuevos. Es muy importante evitar esto ya que el objetivo es que se pueda predecir el precio de un alojamiento en Airbnb en el futuro.

Una vez hecho eso, se entrena el modelo con el sub dataframe train y se evalúa la predicción con el test. Se obtiene una **R<sup>2</sup> de 0,967**, lo cual es un valor muy bueno. Cuanto más se acerque el valor de esta métrica al 1, más se ajusta el modelo. Sin embargo, no tiene en cuenta el **overfitting**. Para ello hay otras métricas que se pueden usar en conjunto con esta, como el error cuadrático medio (MSE).

El **MSE** da un valor que indica cuánto se ajusta el modelo; cuánto se desvían los resultados obtenidos de los números reales. Al dar un valor absoluto, por sí solo no

es muy útil, pero puede usarse para comparar con otros modelos y seleccionar el mejor. Este sería el MSE con valor más cercano a 0. Por lo tanto, para compararlo y para comprobar si se puede mejorar el modelo, se cambiarán los diferentes parámetros que admite la regresión lineal con **GridSearch**. En la tabla inferior se pueden comparar los resultados de ambos modelos.

	R <sup>2</sup>	MSE
Regresión lineal	0,967	33,82
Regresión lineal GS	-34,498	33,82

El objetivo era afinar el modelo para conseguir un mejor resultado, concretamente un mejor MSE, porque el R<sup>2</sup> ya tenía un valor muy alto y se ha de evitar recompensar el overfitting. El objetivo de este modelo es que con los parámetros que se le da pueda predecir futuros precios. Por esa razón hay que evitar que el modelo se ajuste demasiado bien a estos datos. Sin embargo, el **MSE no mejora**, y el R<sup>2</sup> empeora mucho, se vuelve negativo. Un R<sup>2</sup> compara el modelo con una línea horizontal, y si éste predice peor que esa línea, el resultado es negativo. Por lo tanto, este modelo es mucho peor.

#### Pruebas con otros modelos

Viendo la gran correlación entre variables, era de esperar de primeras unos resultados tan altos en cuanto ajuste del modelo. La regresión lineal es también un algoritmo bastante sencillo y tiene poco margen de mejora, y más con estos resultados. Sin embargo, existen otros algoritmos que tal vez puedan crear un mejor modelo. Para compararlos, se usará Pycaret. De igual modo se divide el dataset en train y test, y se transforman los datos para mejorar los modelos y compara los diferentes algoritmos de regresión según las posibles métricas.



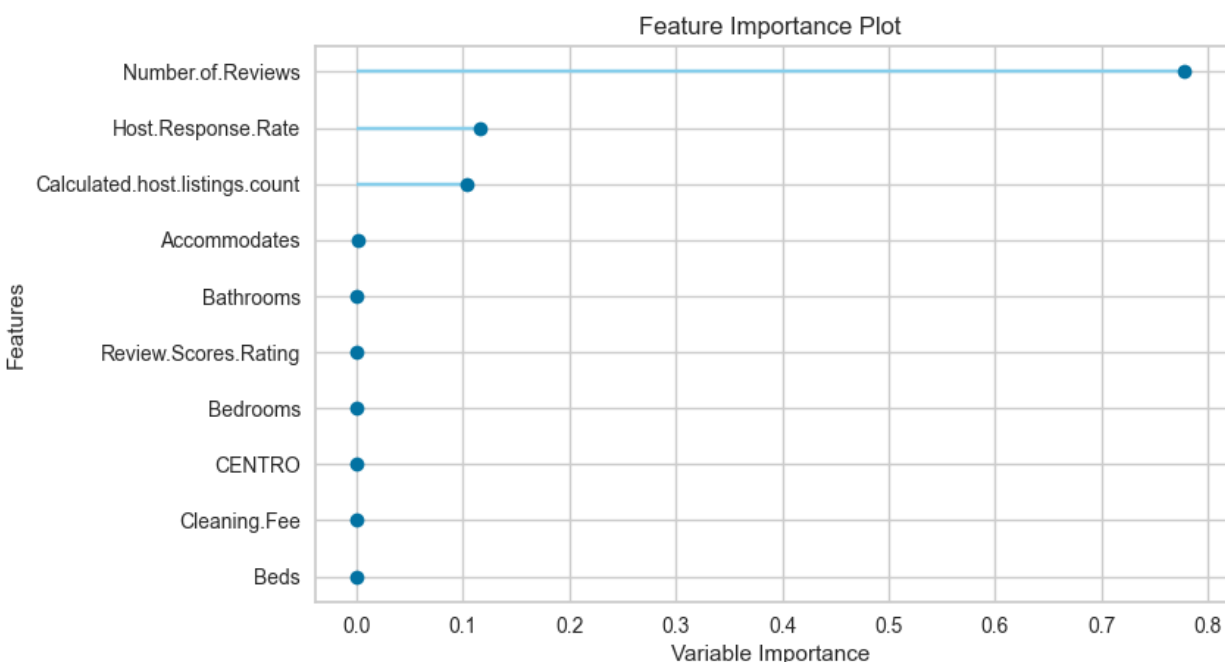
	$R^2$	MSE
<b>Random Forest Regressor</b>	0,9997	0,2915
<b>Decision Tree Regressor</b>	0,9997	0,3361
<b>Light Gradient Boosting Machine</b>	0,9997	0,3299
<b>Extra Trees Regressor</b>	0,9996	0,4108
<b>Gradient Boosting Regressor</b>	0,9995	0,5303

En la tabla superior se pueden ver los resultados obtenidos por los cinco mejores modelos en la comparación. Hay tres modelos que dan el mismo resultado en cuanto a  $R^2$ , pero el modelo **Random Forest obtiene el mejor MSE**, y es dos órdenes de magnitud mejor que el modelo de regresión lineal. En el notebook `model.ipynb` se puede encontrar la tabla completa con los 17 modelos y otras métricas, en las que también sobresale el modelo de Random Forest.


A continuación, se hace un GridSearch de nuevo para afinar este modelo, y se entrena con todos los datos, incluyendo todo el dataset. De esta manera, se ajustará mejor a nuevos datos en el futuro. Se obtienen métricas similares, descartando el overfitting de nuevo. En el cuadro inferior se puede ver la comparación del modelo definitivo con los dos modelos de regresión lineal.

	$R^2$	MSE
<b>Regresión lineal</b>	0,967	33,82
<b>Regresión lineal GS</b>	-34,498	33,82
<b>Random Forest</b>	0,9997	0,2924

El modelo de regresión de **Random Forest** es el claro ganador. Es un algoritmo formado por varios Decision Trees, otro tipo de algoritmo de machine learning. Éste consiste en ir separando los datos mediante preguntas o divisiones con respuesta binaria, y pueden tender a causar overfitting o sesgos. Este problema se puede evitar con múltiples Decision Trees, aumentando la precisión en los resultados. En este caso, el algoritmo contiene 100 Decision Trees.



Otra ventaja de este modelo es que permite evaluar fácilmente la contribución de las variables en el modelo. Cabe aclarar que no tiene que ver con la correlación, es tan solo los parámetros que ha tenido en cuenta el algoritmo a la hora de calcular. Se puede observar que el número de reseñas de un alojamiento tiene la mayor



importancia con diferencia, seguido por la tasa de respuesta del anfitrión y el número de anuncios del anfitrión en la página web.

Por último, se guarda el modelo para poder consultarlo más adelante en un archivo pickle, con extensión .pkl. Este tipo de archivos permite serializar un objeto de Python, como puede ser la pipeline del modelo, para que pueda ser guardada como un archivo en el disco del ordenador.

## Suposiciones iniciales.

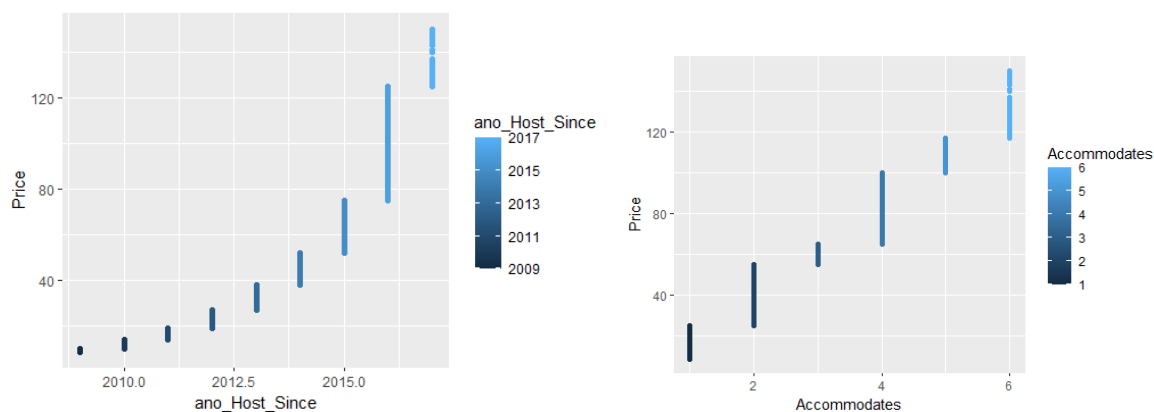
Los precios de los alojamientos situados en el Centro son mayores que los del resto de los barrios.	El precio NO es mayor en los alojamientos situados en el Centro.
Los precios de los alojamientos que pertenecen a un host que lleva más tiempo serán más altos.	Al contrario, un host que lleva más tiempo tendrá alojamientos más baratos.
El precio aumentará según el: <ul style="list-style-type: none"> <li>• número de huéspedes.</li> <li>• número de reviews</li> <li>• Score rating</li> <li>• Host response rate</li> <li>• habitaciones</li> <li>• baños</li> <li>• camas</li> </ul>	El precio aumentará según el: <ul style="list-style-type: none"> <li>• número de huéspedes.</li> <li>• número de reviews</li> <li>• habitaciones</li> <li>• baños</li> <li>• camas</li> </ul> El precio NO aumentará según el: <ul style="list-style-type: none"> <li>• Score rating</li> <li>• Host response rate</li> </ul>
El precio cuando el tipo de propiedad es "Casa"	El precio del tipo de propiedad "Casa" no es mayor que el de otros tipos de propiedad.
El precio es superior cuando si el tipo de habitación es "Entero"	El precio no es superior al de otros tipos de habitación.
El precio aumenta cuando el tipo cama es "Cama Real"	El precio no aumenta si el tipo de cama es "Real"
El precio de aquellos alojamientos que piden fianza o tasa de limpieza es superior a los que no.	El precio de aquellos alojamientos que piden fianza no son superiores.
El precio aumentará si el host tiene más alojamientos alquilados.	El precio no aumenta si el host no tiene más alojamientos alquilados.

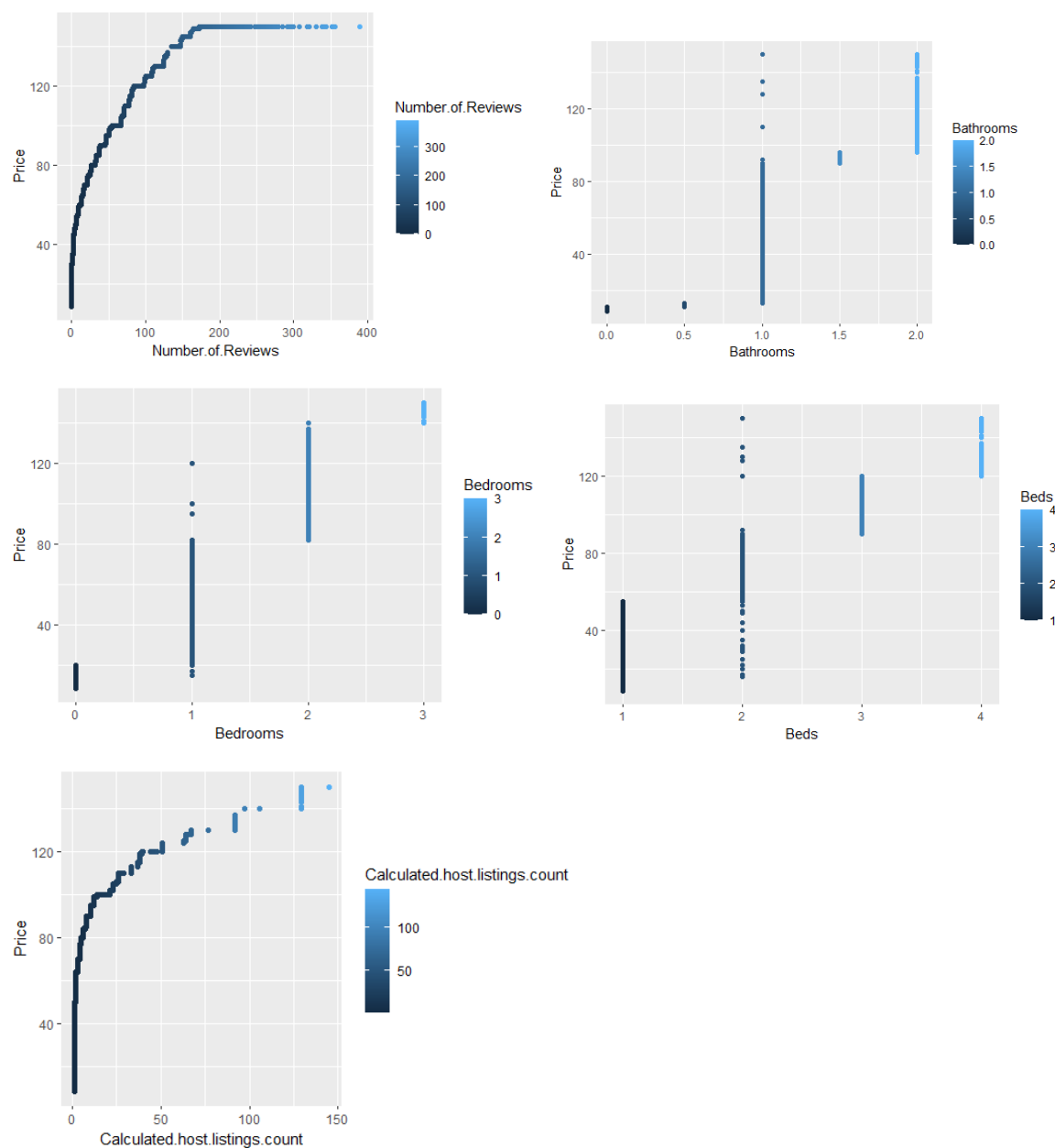
## Conclusiones.

Finalmente, analizando la matriz de correlaciones se puede concluir que las variables con más correlación son:

- Host Since
- Accommodates
- Number of reviews
- Bathrooms
- Bedrooms
- Beds
- Calculated.host.listings.count

La representación gráfica, para ver la relación de cada una de las variables con la variable price es la siguiente:





La organización del grupo en cuanto a repartición de tiempo y tareas fue uno de los puntos fuertes. A pesar de la falta de experiencia con metodologías ágiles, nos supimos adaptar bien a la nueva forma de trabajar. Nos permitió organizar bien el tiempo y ayudó cuando tuvimos imprevistos. Sin duda en el futuro continuaremos usando Kanban.

Las reuniones regulares a medida que se iban completando los diferentes hitos nos ayudó a llevar mejor control de cómo avanzábamos y dónde nos quedábamos rezagadas. A continuación se muestra un calendario de nuestras actividades para completar este proyecto.



Puesto que teníamos un tiempo tan limitado, nuestra prioridad era conseguir sacar un proyecto que funcionara. Por tanto, en algunos puntos en los que se puede mejorar técnicamente. El código se puede pulir, y hay algunas variables que descartamos por falta de tiempo, pero que con más recursos se podrían usar para extraer datos interesantes.

Un ejemplo son las columnas con datos como comentarios o descripciones, que son difíciles de normalizar, pero que con Natural Language Processing se podrían analizar. Nuestra propuesta sería realizar un sentiment analysis y ver si podemos probar una relación con el precio, nuestro KPI principal.