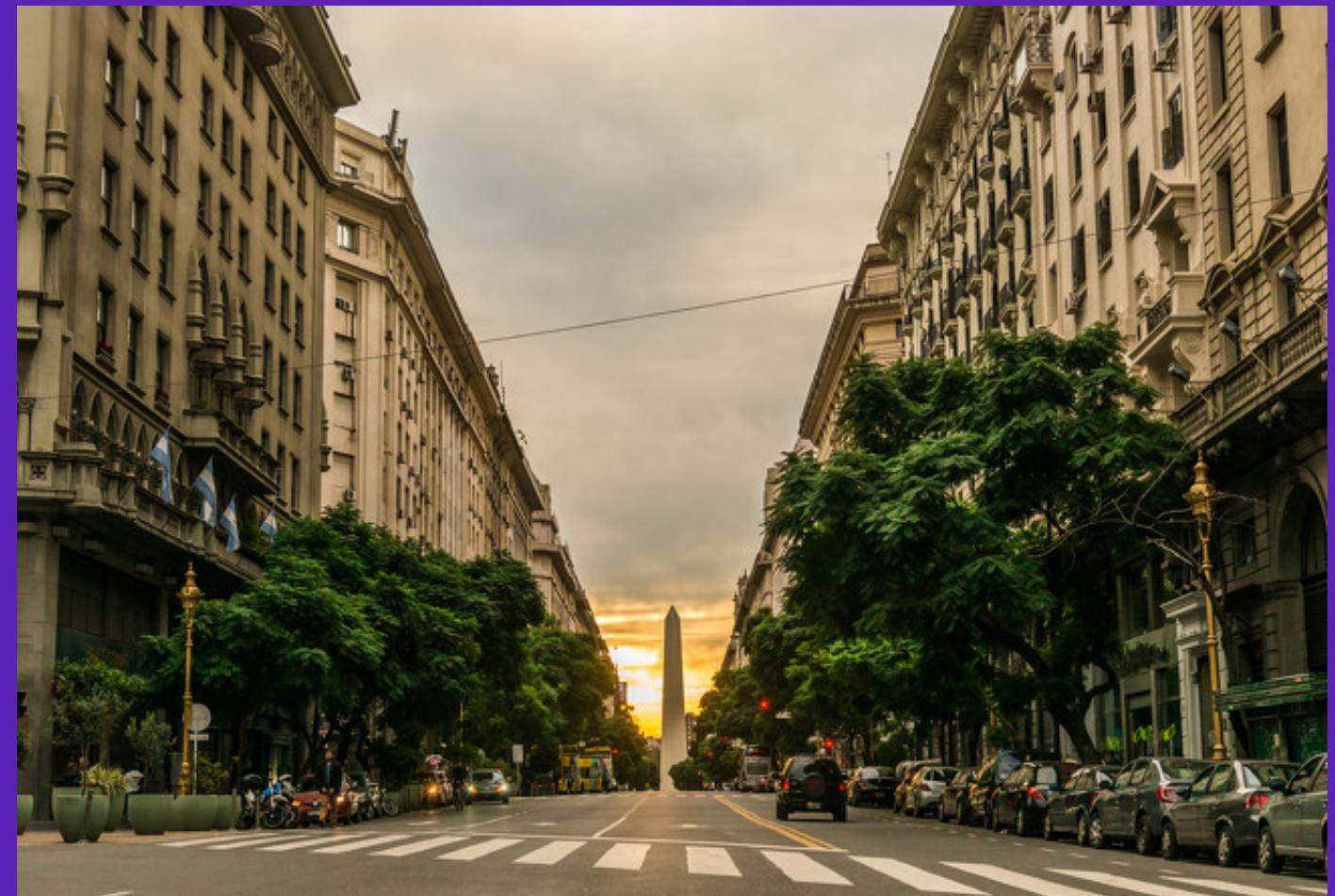


¿Cómo predecir el precio de una propiedad? Machine learning con datos abiertos

Objetivo

Nos proponemos construir un modelo que prediga el precio de las propiedades en la Ciudad de Buenos Aires (CABA), la capital de Argentina.

Para ello, pensamos en qué información de acceso público podría ayudarnos a la hora de proponer un modelo predictivo exitoso.



Queremos usar datos reales para obtener resultados reales

Etapas del proceso de trabajo

Paso 1: Formular una pregunta

Nos preguntamos qué variables debería tener un modelo para poder hacer una buena predicción del precio de una propiedad en CABA

Paso 2: Limpiar el dataset principal

Limpiamos el dataset y seleccionamos las columnas que queríamos usar

Paso 3: Analizar el dataset principal

Exploramos distintas columnas del dataset principal

Paso 4: Aregar columnas de otros datasets

Elegimos datasets que creemos que tienen variables que las personas tienen en cuenta al decidir cuánto gastar en una propiedad.

Paso 5: Correr los modelos y obtener resultados

Usamos Regresión Linear, Random Forest Regressor y XGBoost Regressor y obtuvimos sus resultados.

Paso 6: Conclusiones, consideracion es y próximos pasos

Analizamos los resultados y obtuvimos las conclusiones. Hicimos aclaraciones y pasos que quedaron pendientes.

Nuestra pregunta:

¿Qué variables pueden ayudar a predecir el precio de una propiedad en la Ciudad de Buenos Aires?



Dataset principal

Con el propósito mencionado, nuestro principal dataset fue el de Properati (<https://www.properati.com.ar/data>).

En este sitio puede descargar los datos de más de 2 millones de propiedades de Argentina, Colombia, Ecuador y Perú.

Para este proyecto, usamos un conjunto de datos con propiedades de los últimos 6 meses.

Paso 1: Formular una
pregunta

1

Dataset principal

Properati Dataset

<https://www.properati.com.ar/data>

```
properties = pd.read_csv('datasets/ar_properties.csv')
```

```
properties.shape
```

```
(1000000, 25)
```

Dataset principal

id - Identificador del aviso.

ad_type - Tipo de aviso (Propiedad, Desarrollo/Proyecto).

start_date - Fecha de alta del aviso.

end_date - Fecha de baja del aviso.

created_on - Fecha de alta de la primera versión del aviso.

lat - Latitud.

lon - Longitud.

11 - Nivel administrativo 1: país.

12 - Nivel administrativo 2: usualmente provincia.

13 - Nivel administrativo 3: usualmente ciudad.

14 - Nivel administrativo 4: usualmente barrio.

rooms - Cantidad de ambientes (útil en Argentina).

bedrooms - Cantidad de dormitorios (útil en el resto de los países).

bathrooms - Cantidad de baños.

surface_total - Superficie total en m².

surface_covered - Superficie cubierta en m².

price - Precio publicado en el anuncio.

currency - Moneda del precio publicado.

price_period - Periodo del precio (Diario, Semanal, Mensual).

title - Título del anuncio.

description - Descripción del anuncio.

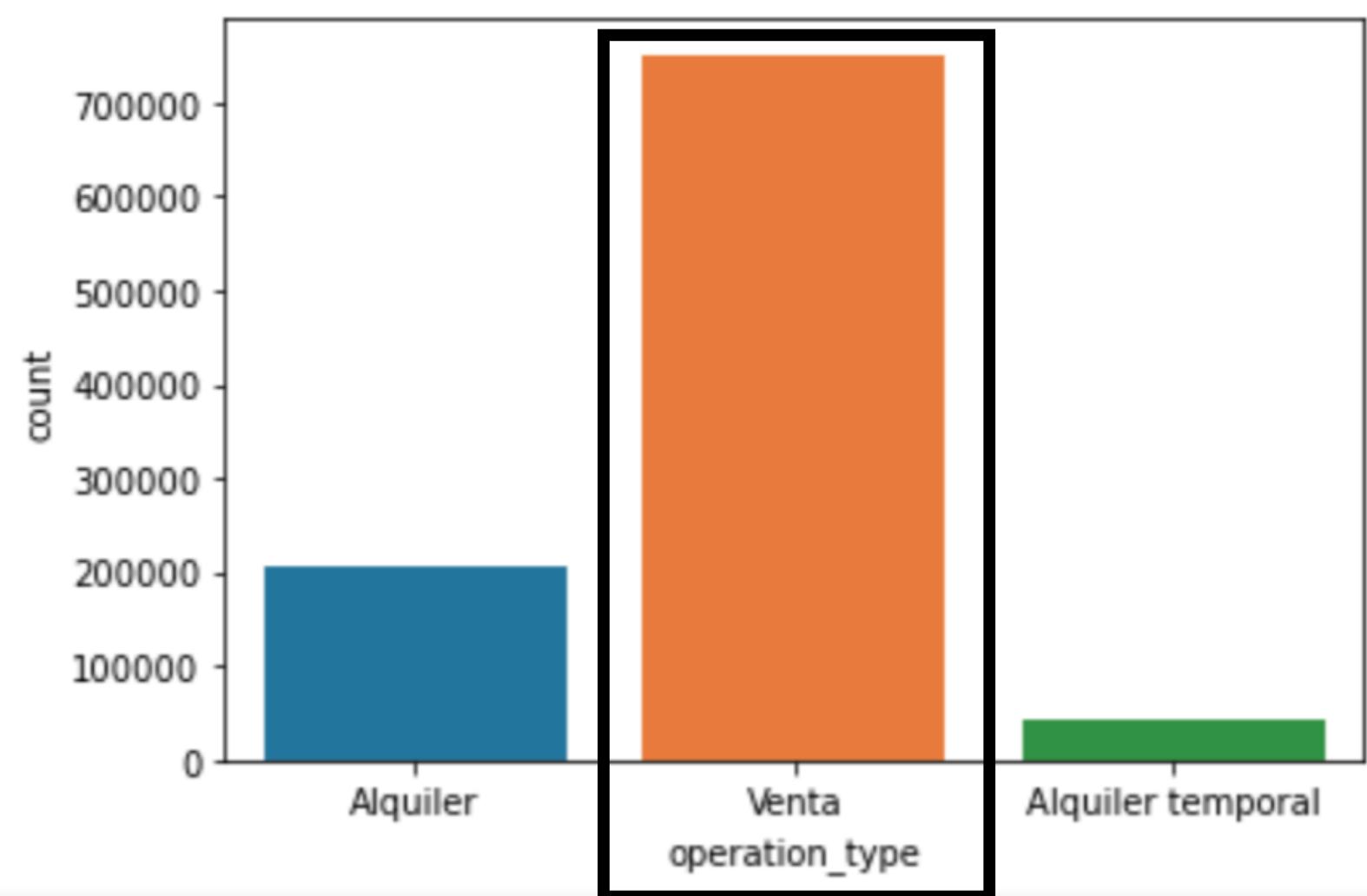
operation_type - Tipo de operación (Venta, Alquiler, Alquiler Temporario).

property_type - Tipo de propiedad (Casa, Departamento, PH).

Limpieza

- 1) Elegimos solamente las propiedades en venta ("venta")

```
properties_sale = properties[properties['operation_type'] == "Venta"]
properties_sale.shape
(750607, 25)
```



Paso 2: Limpiar el dataset principal

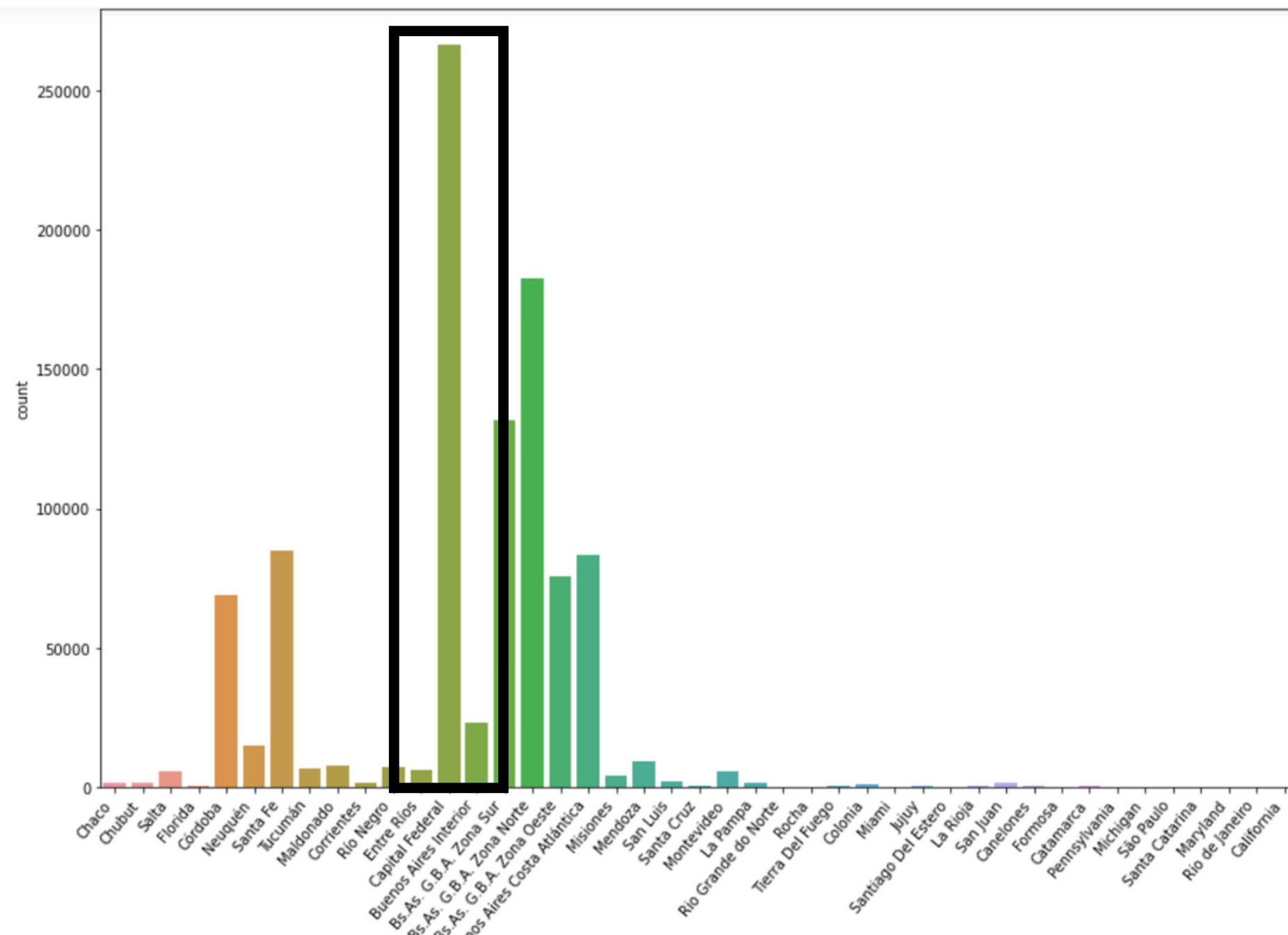
2

Limpieza

2) Elegimos solamente las propiedades ubicadas en la Ciudad de Buenos Aires

```
properties_sale_CABA = properties_sale[properties_sale['l2'] == "Capital Federal"]  
properties_sale_CABA.shape
```

(185481, 25)

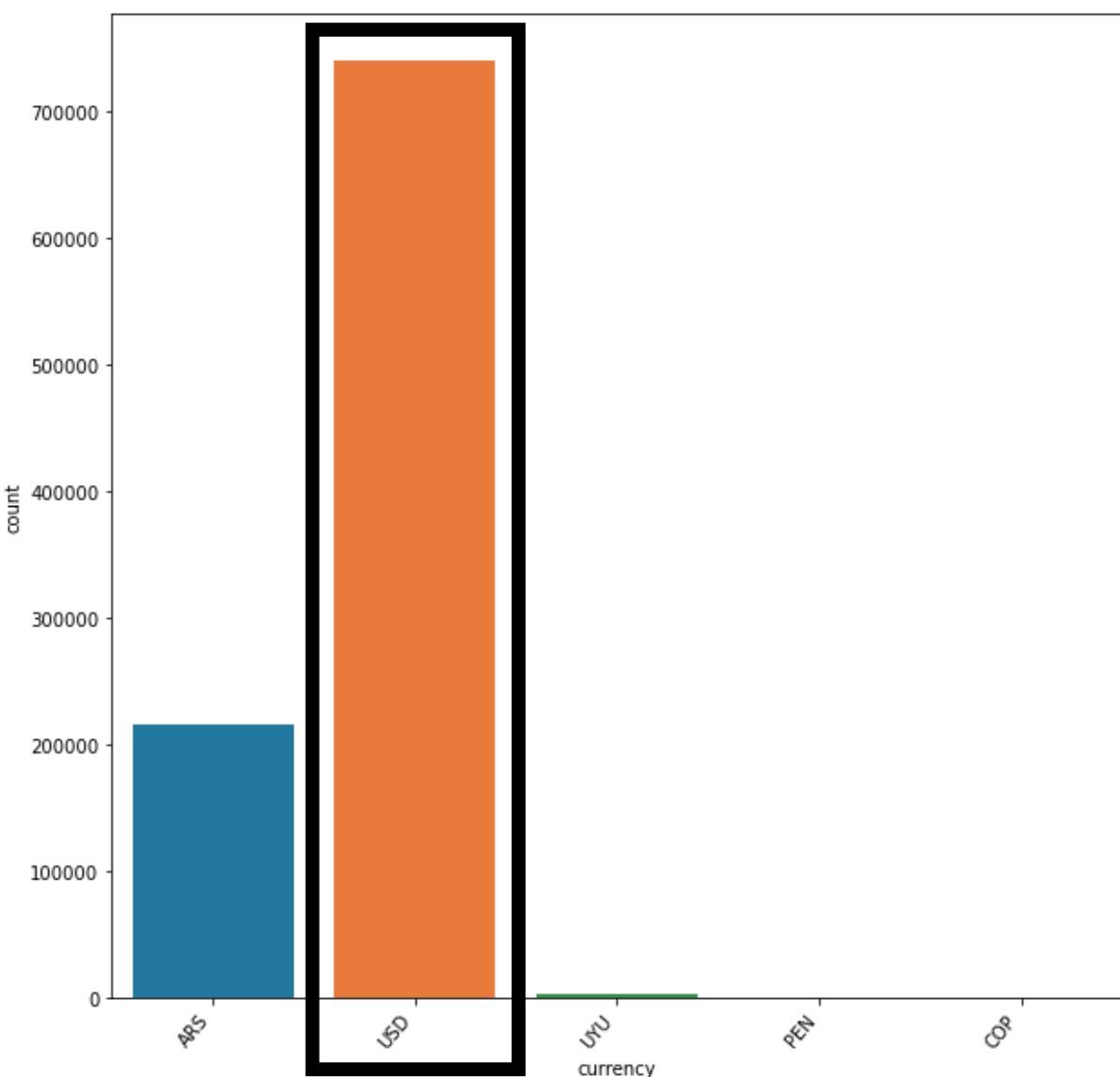


Limpieza

- 3) Elegimos solamente las propiedades en USD

```
properties_sale_CABA = properties_sale_CABA[properties_sale_CABA['currency'] == "USD"]  
properties_sale_CABA.shape
```

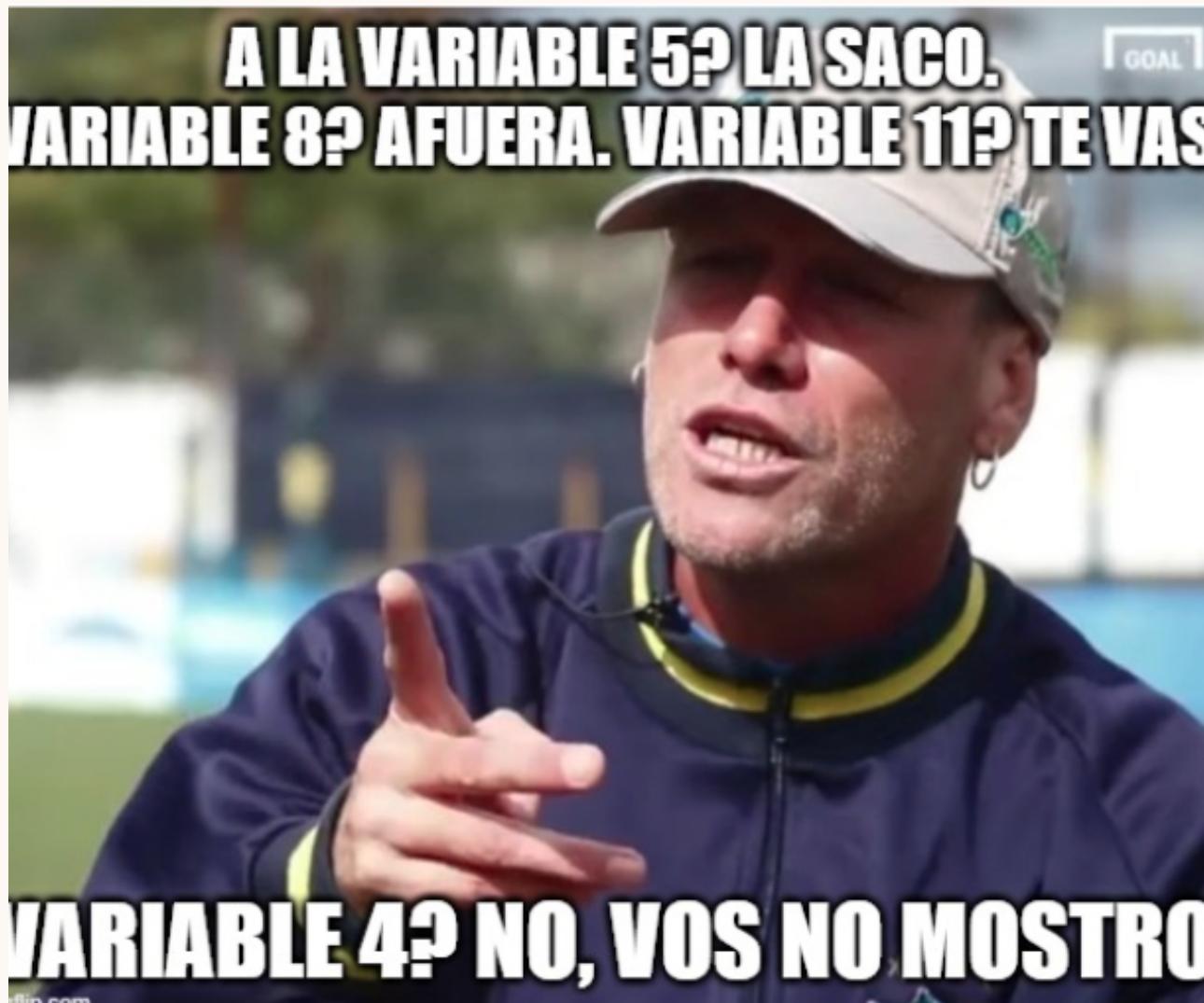
(181303, 25)

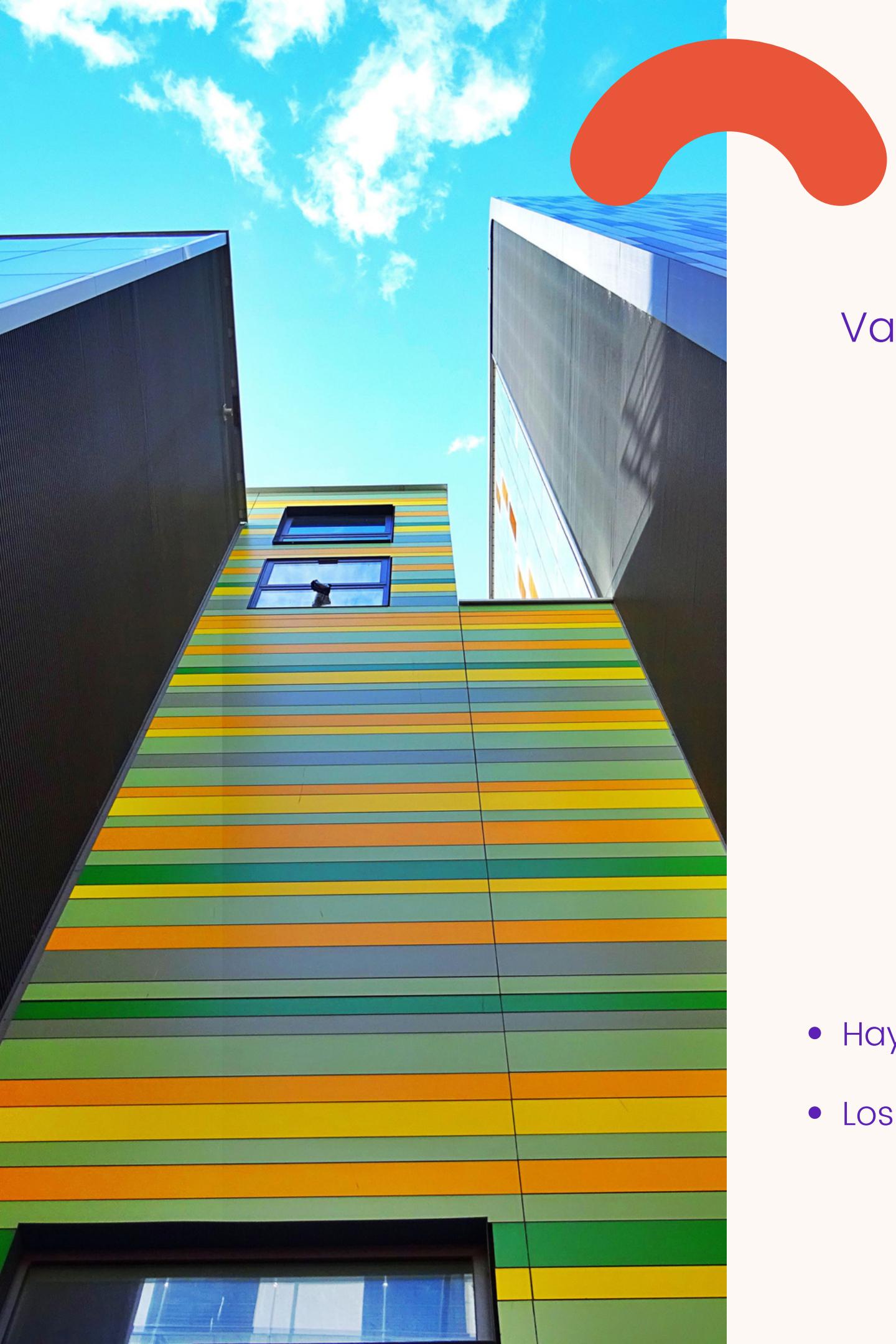


Limpieza

- 4) Seleccionamos las columnas que vamos a utilizar

```
properties_sale_CABA.drop(['id','ad_type','l1','l2','l4','l5','l6',
'currency','price_period','title','description','operation_type','created_on', 'bedrooms'], axis=1, inplace=True)
properties_sale_CABA.shape
(181303, 11)
```





Limpieza

Variables seleccionadas:

- latitud
 - longitud
 - barrios
 - ambientes
 - baños
 - superficie total
 - superficie cubierta
 - precio
 - tipo de propiedad
 - fecha de alta del aviso
 - fecha de baja del aviso
-
- Hay 48 barrios.
 - Los tipos de propiedad son:

PH	Lote
Departamento	Depósito
Local comercial	Cochera
Oficina	Casa de campo
Casa	Otro

Paso 2: Limpiar el dataset principal

2

Limpieza

- 5) Unificamos los nombres de los barrios: reemplazo los barrios "falsos" por los "verdaderos".

Por ejemplo: Once no existe como barrio, es Balvanera, Las Cañitas es Palermo, etc.

```
properties_sale_CABA.replace('Parque Centenario', 'Caballito', inplace=True)
properties_sale_CABA.replace('Once', 'Balvanera', inplace=True)
properties_sale_CABA.replace('Catalinas', 'Boca', inplace=True)
properties_sale_CABA.replace('Las Cañitas', 'Palermo', inplace=True)
properties_sale_CABA.replace('Congreso', 'Balvanera', inplace=True)
properties_sale_CABA.replace('Tribunales', 'San Nicolas', inplace=True)
properties_sale_CABA.replace('Centro / Microcentro', 'San Nicolas', inplace=True)
properties_sale_CABA.replace('Abasto', 'Almagro', inplace=True)
properties_sale_CABA.replace('Barrio Norte', 'Recoleta', inplace=True)
```

Limpieza

6) Manejo de NaNs y de datos inconsistentes

Buscamos la cantidad de NaNs que tiene el dataset por columna

```
properties_sale_CABA.isna().sum()
```

start_date	0
end_date	0
lat	11935
lon	11964
neighbourhood	2356
rooms	29213
bathrooms	24285
surface_total	62437
surface_covered	64078
price	0
property_type	0
dtype:	int64



0 vs NULL

Nan: Not a Number, representan un valor vacío

Limpieza

6) Manejo de NaNs y de datos inconsistentes

Vemos que hay muchas variables con valores NaN, así que creamos "reglas" para manejar este problema:

- Eliminamos los valores que tienen los atributos "lat" y "lon" como NaN.

```
properties_sale_CABA.dropna(subset=['lat', 'lon'], axis=0, inplace=True)  
properties_sale_CABA.shape
```

(169339, 11)

- Si el atributo 'rooms' es NaN, entonces eliminamos la fila completa.

```
properties_sale_CABA.dropna(subset=['rooms'], axis=0, inplace=True)  
properties_sale_CABA.shape
```

(142554, 11)

Limpieza

6) Manejo de NaNs y de datos inconsistentes

- Si los atributos 'surface_total' y 'surface_covered' son ambos NaN, entonces eliminamos la fila completa.

```
properties_sale_CABA.dropna(subset=['surface_total', 'surface_covered'], how='all', axis=0, inplace=True)  
properties_sale_CABA.shape  
(99498, 11)
```

- Si los atributos 'surface_total' o 'surface_covered' son NaN, entonces llenamos la columna NaN con los datos de la otra

```
properties_sale_CABA['surface_total'].fillna(properties_sale_CABA['surface_covered'], inplace=True)  
properties_sale_CABA['surface_covered'].fillna(properties_sale_CABA['surface_total'], inplace=True)
```

- Si el atributo 'surface_covered' es mayor a 'surface_total', los intercambio

```
properties_sale_CABA['surface_covered'], properties_sale_CABA['surface_total'] = \  
    np.where(properties_sale_CABA['surface_covered'] > properties_sale_CABA['surface_total'], \  
            (properties_sale_CABA['surface_total'], properties_sale_CABA['surface_covered']), \  
            (properties_sale_CABA['surface_covered'], properties_sale_CABA['surface_total']))
```

Limpieza

6) Manejo de NaNs y de datos inconsistentes

- Si el atributo 'neighbourhood' es NaN, entonces eliminamos la fila completa

```
properties_sale_CABA.dropna(subset=['neighbourhood'], axis=0, inplace=True)  
properties_sale_CABA.shape  
(99189, 11)
```

- Si el atributo 'bathrooms' es NaN, entonces le ponemos 1.

```
properties_sale_CABA[['bathrooms']] = properties_sale_CABA[['bathrooms']].fillna(1)
```

Limpieza

6) Manejo de NaNs y de datos inconsistentes

```
properties_sale_CABA.isnull().sum()
```

```
start_date      0
end_date       0
lat            0
lon            0
neighbourhood 0
rooms          0
bathrooms      0
surface_total   0
surface_covered 0
price          0
property_type   0
dtype: int64
```

Ya no quedan datos no definidos.

Dataset principal

Agregamos una variable más llamada "tiempo_en_venta", que es la diferencia entre el end_date y el start_date
El dataset quedaría con 87.146 datos y 10 columnas.

```
properties_sale_CABA.head(5)
```

	lat	lon	neighbourhood	rooms	bathrooms	surface_total	surface_covered	price	property_type	tiempo_en_venta
92	-34.603771	-58.381587	Villa Crespo	7.0	2.0	130.0	130.0	320000.0	PH	15
124	-34.601366	-58.424303	Almagro	1.0	1.0	49.0	44.0	90000.0	Departamento	15
125	-34.601366	-58.424303	Almagro	1.0	1.0	49.0	44.0	80000.0	Departamento	15
126	-34.601366	-58.424303	Almagro	1.0	1.0	49.0	44.0	83000.0	Departamento	15
127	-34.601366	-58.424303	Almagro	1.0	1.0	49.0	44.0	84000.0	Departamento	15

Dataset principal

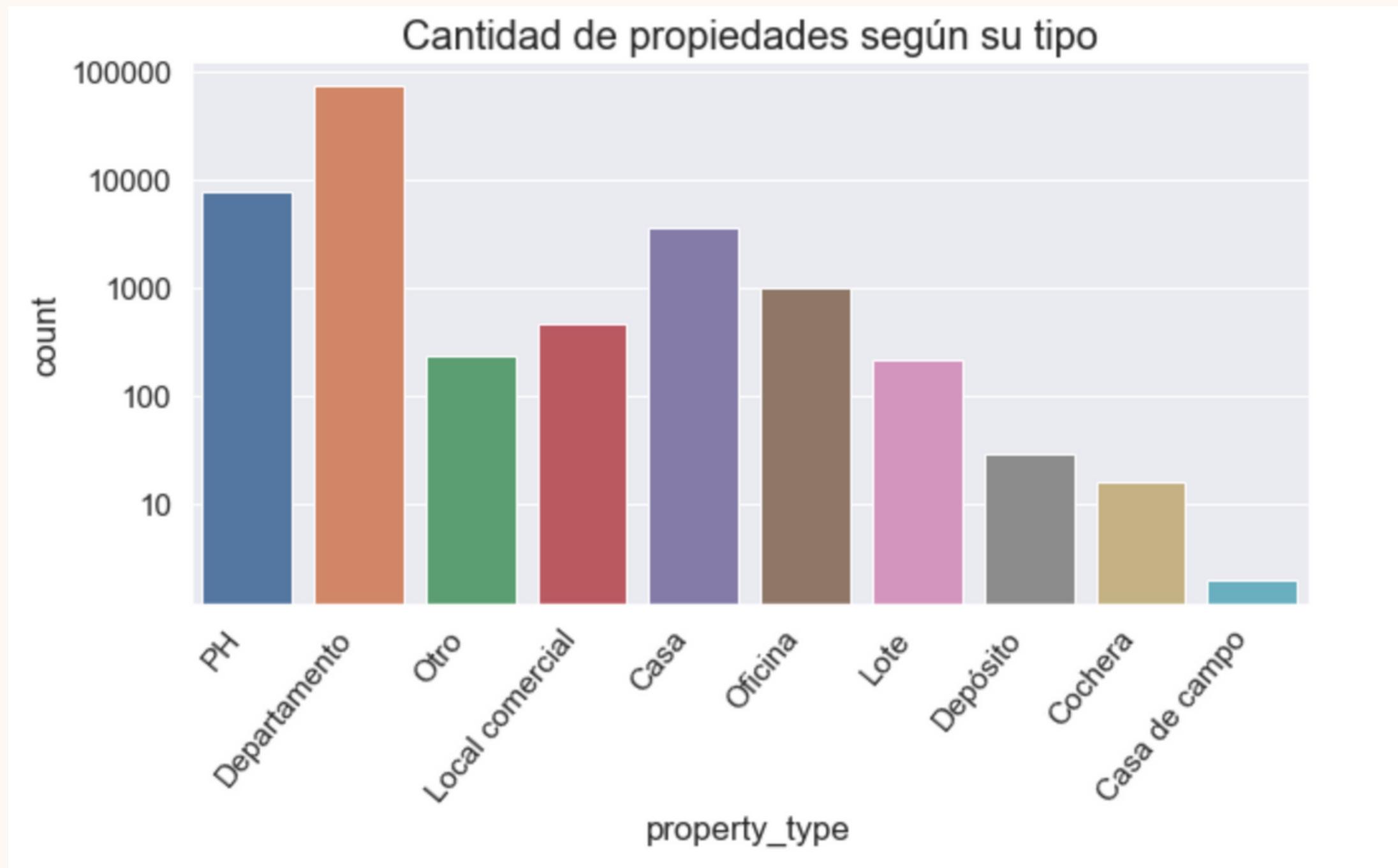
Observamos la información estadístico de nuestro dataset

	lat	lon	rooms	bathrooms	surface_total	surface_covered	price	tiempo_en_venta
count	87101.00000	87101.00000	87101.00000	87101.00000	87101.00000	87101.00000	87101.00000	87101.00000
mean	-34.59571	-58.43723	2.77798	1.47831	122.22441	94.71838	244836.97966	58.57615
std	0.02544	0.03733	1.52558	0.82329	1245.77477	860.83458	326350.18719	69.43318
min	-34.69899	-58.53092	1.00000	1.00000	1.00000	1.00000	5000.00000	0.00000
25%	-34.61322	-58.46337	2.00000	1.00000	45.00000	41.00000	105000.00000	10.00000
50%	-34.59558	-58.43721	3.00000	1.00000	66.00000	59.00000	155000.00000	29.00000
75%	-34.57819	-58.40981	4.00000	2.00000	106.00000	90.00000	255000.00000	85.00000
max	-34.53596	-58.34395	40.00000	17.00000	170000.00000	124370.00000	15011160.00000	437.00000

Dataset principal



Observamos qué tipo de propiedades se encuentran en venta en CABA y cuántas son



Paso 3: Analizar el dataset principal

3

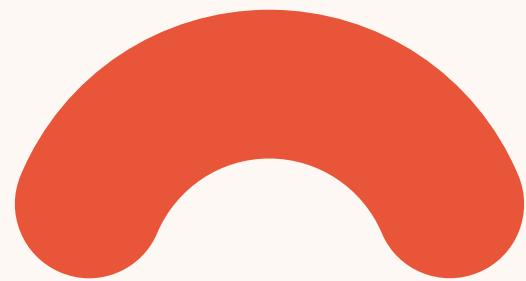
Dataset principal

Eliminamos los elementos correspondientes a "Depósito" y "Cochera", que no poseen ni habitaciones ni baños, y contaminan los datos para el análisis que queremos hacer.

```
properties_sale_CABA = properties_sale_CABA[properties_sale_CABA['property_type'] != 'Depósito']
properties_sale_CABA = properties_sale_CABA[properties_sale_CABA['property_type'] != 'Cochera']
properties_sale_CABA.shape
```

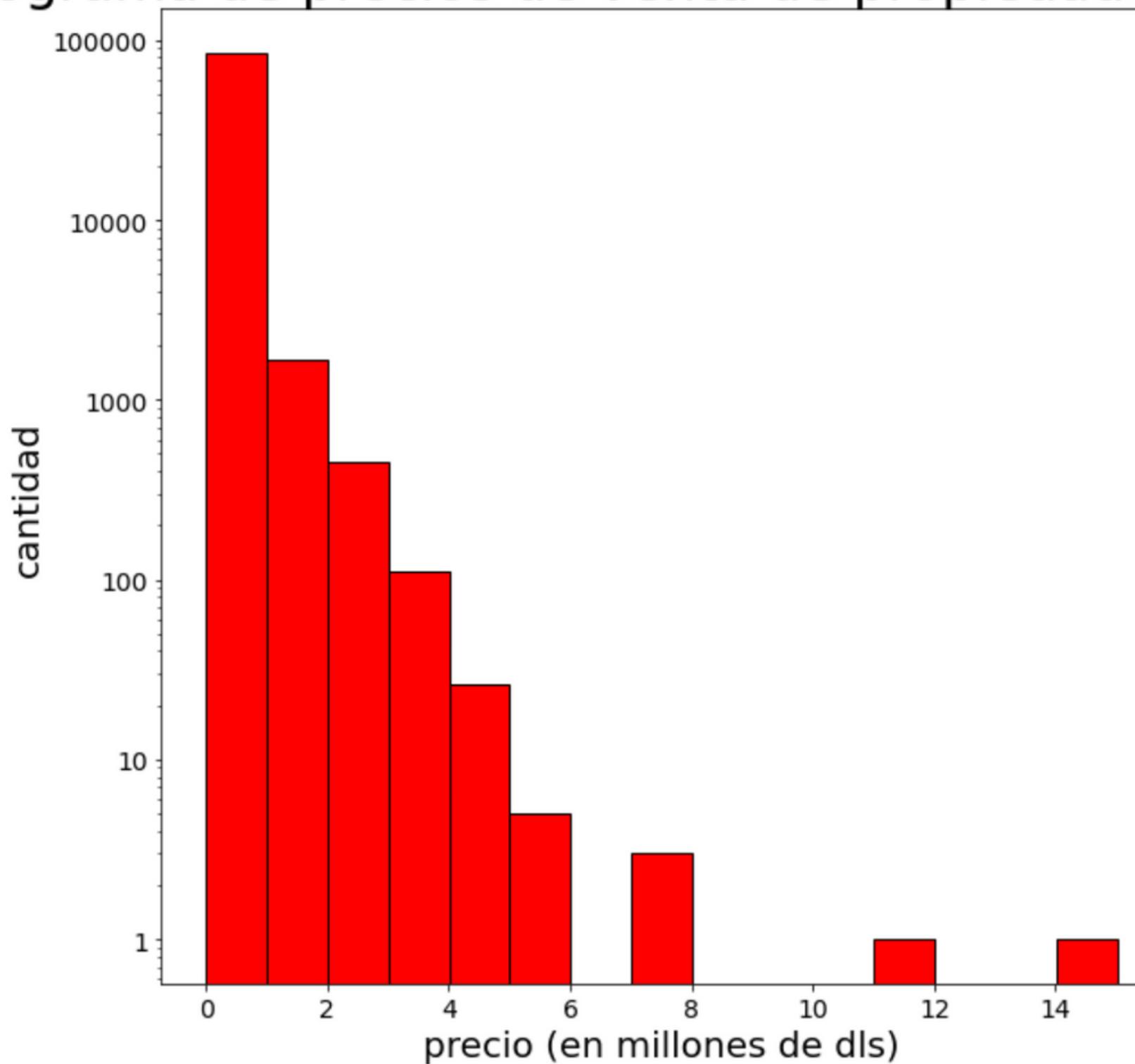
(87101, 10)

Dataset principal



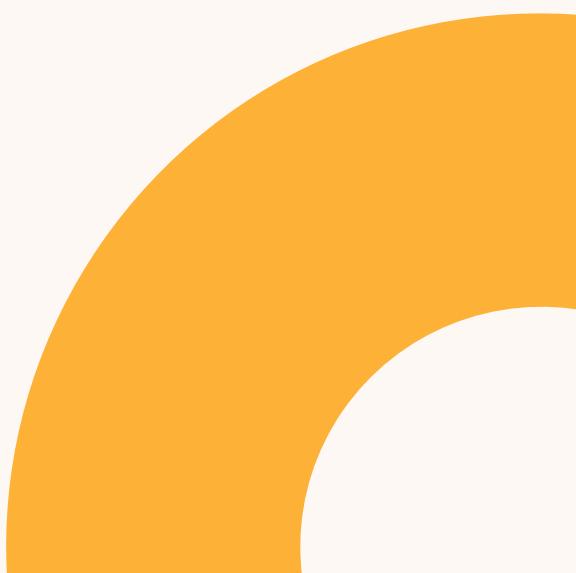
Observamos
cómo se
distribuyen los
precios de venta
de las
propiedades en
CABA

Histograma de precios de venta de propiedades en CABA



Paso 3: Analizar el dataset principal

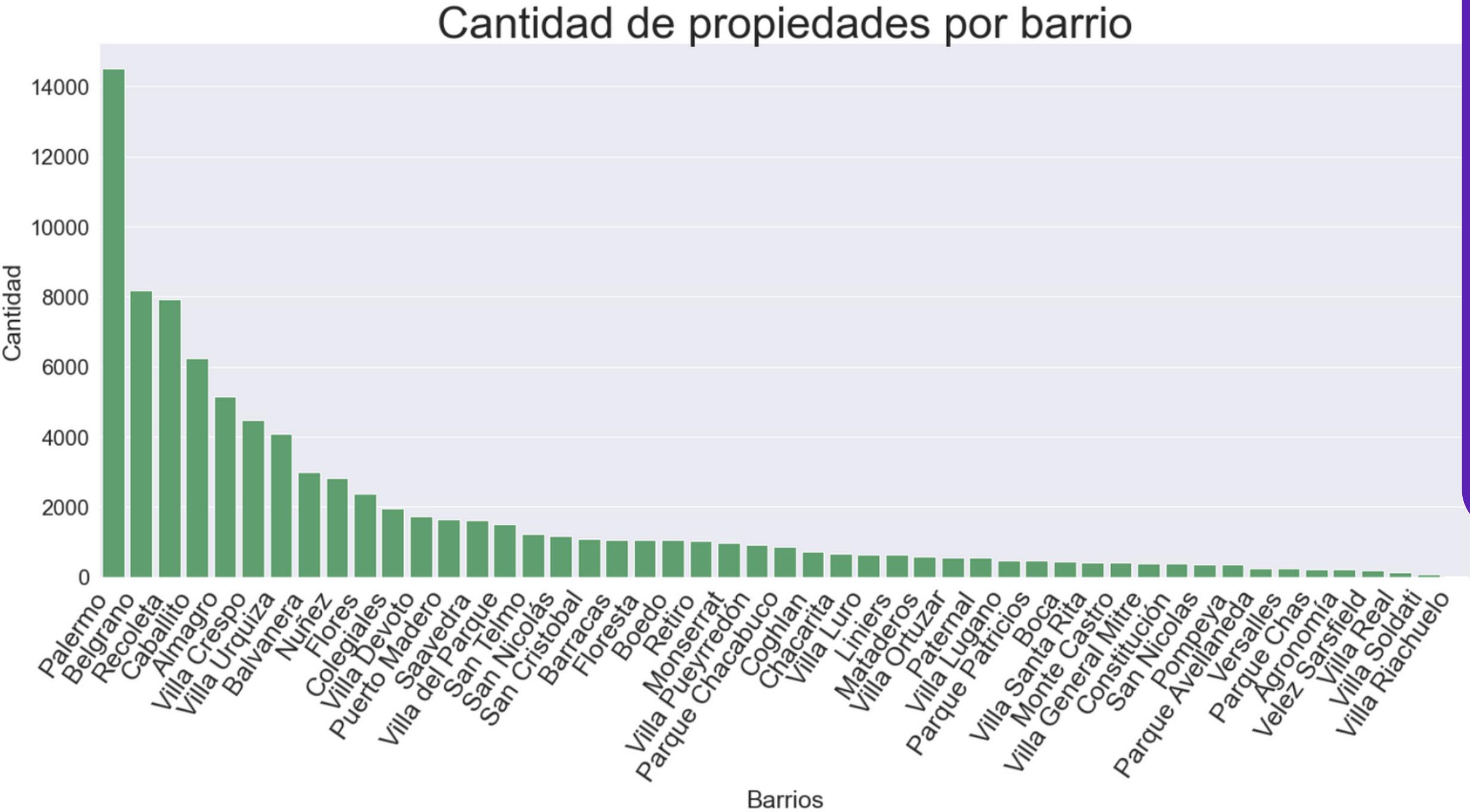
3



Dataset principal



Observamos cuántas propiedades en venta hay en cada barrio

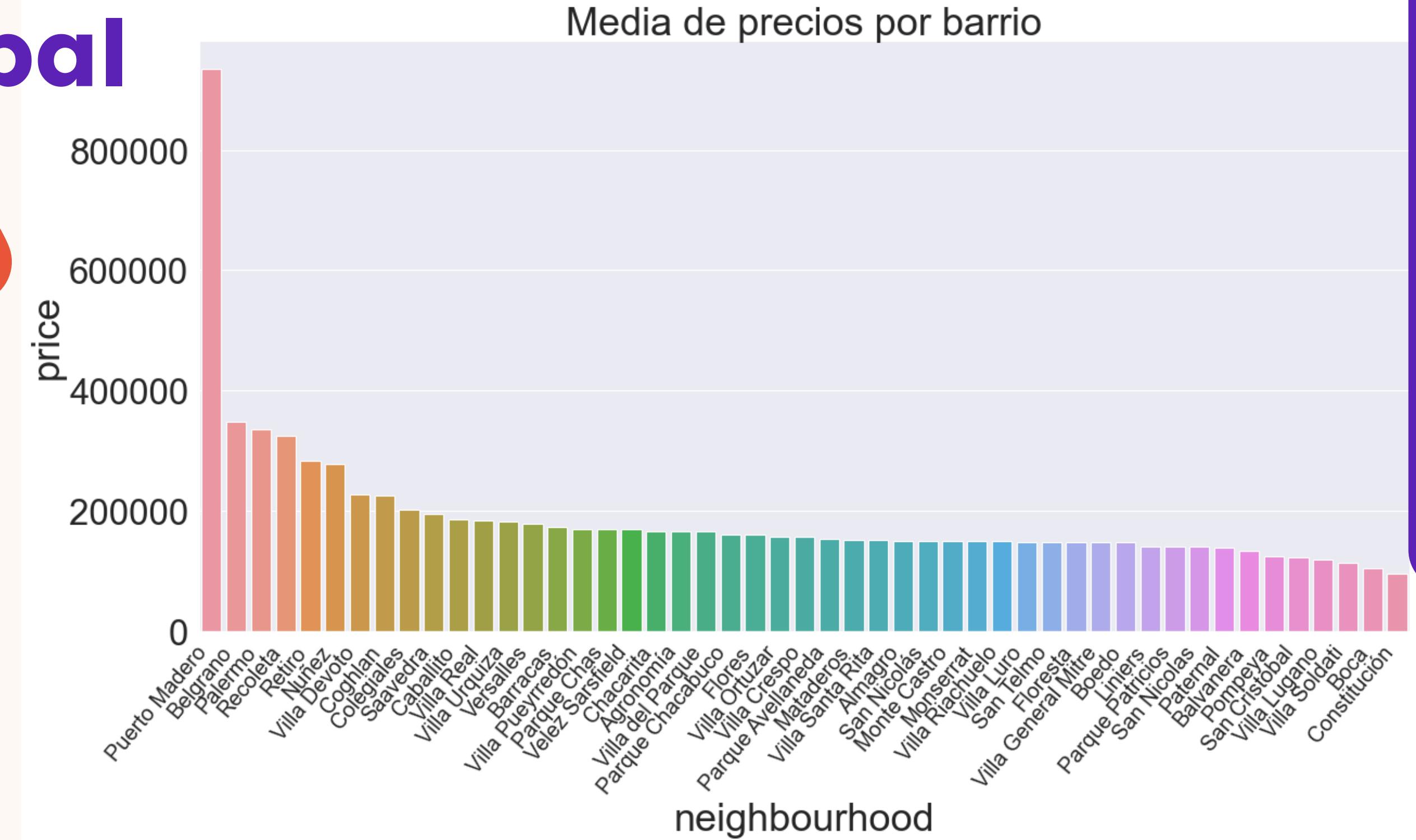


Paso 3: Analizar el dataset principal

3

Dataset principal

Observamos la media de precios por barrio



Paso 3: Analizar el dataset principal

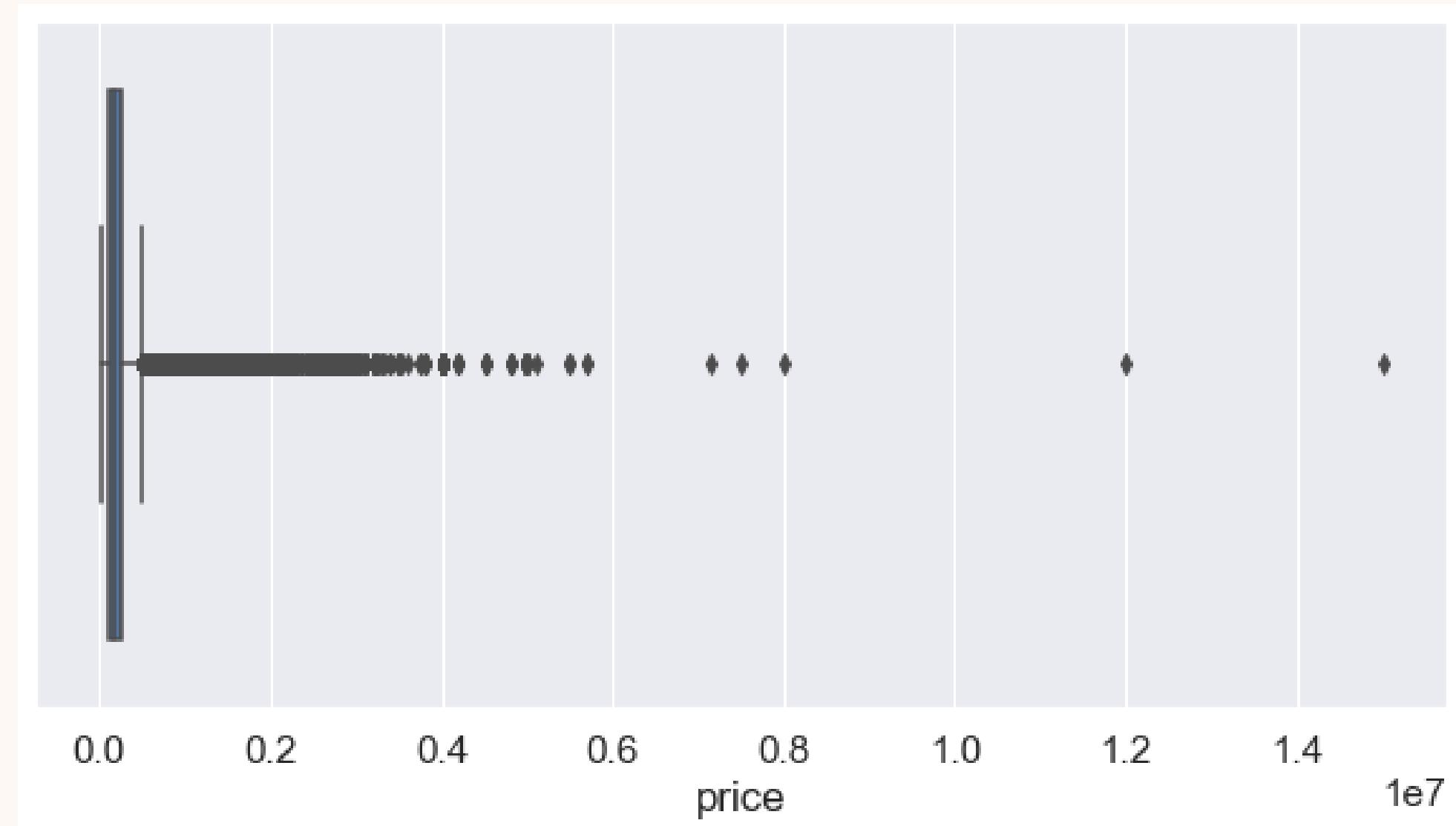
3

Dataset principal



Investigamos si
hay outliers

Boxplot de precios



Vemos 5 puntos que están más lejos que el resto, así que los eliminamos para disminuir la variabilidad

Paso 3: Analizar el dataset principal

3

Dataset principal



Percentiles de precios

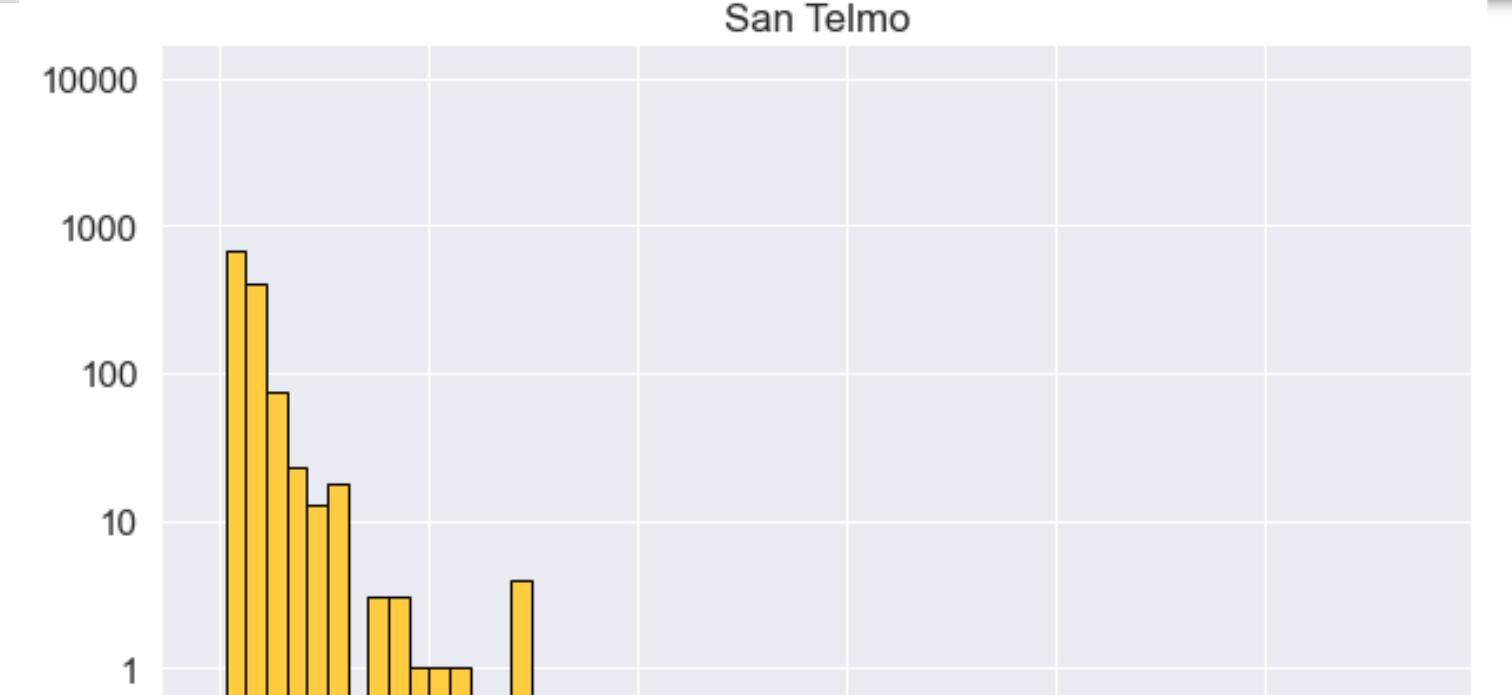
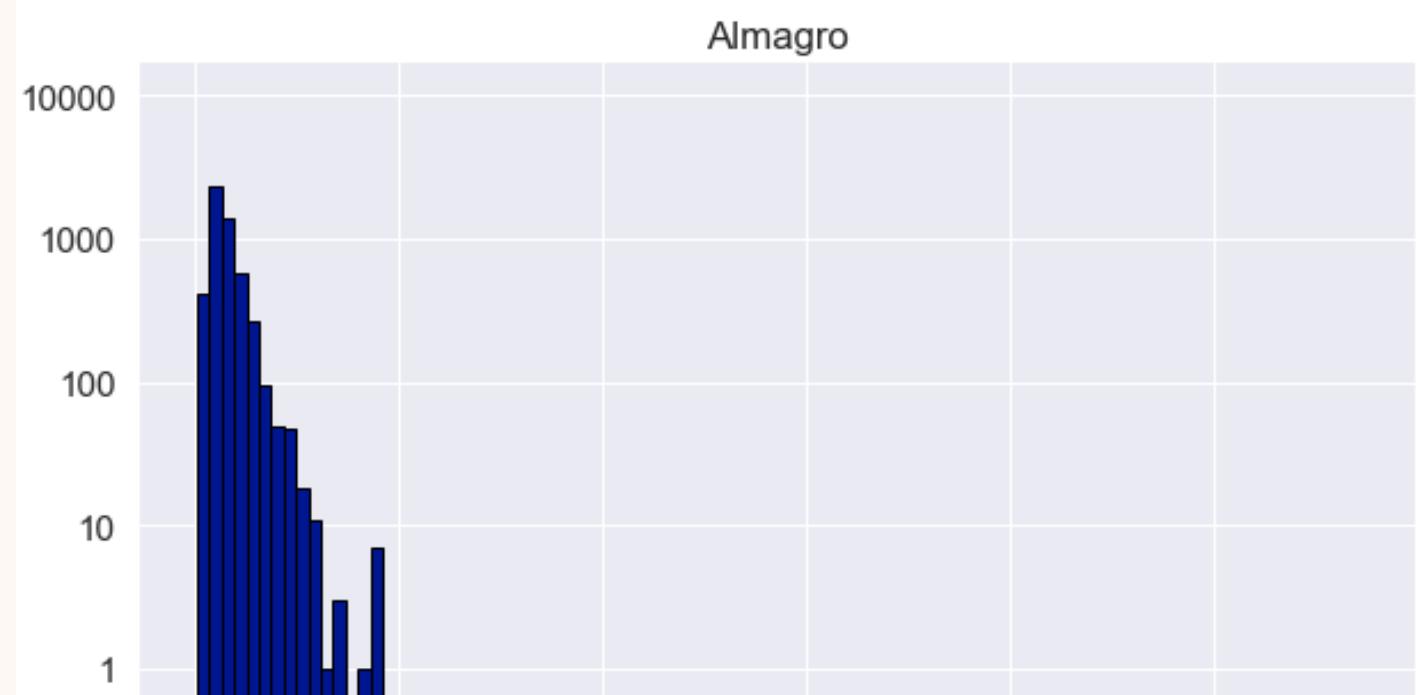
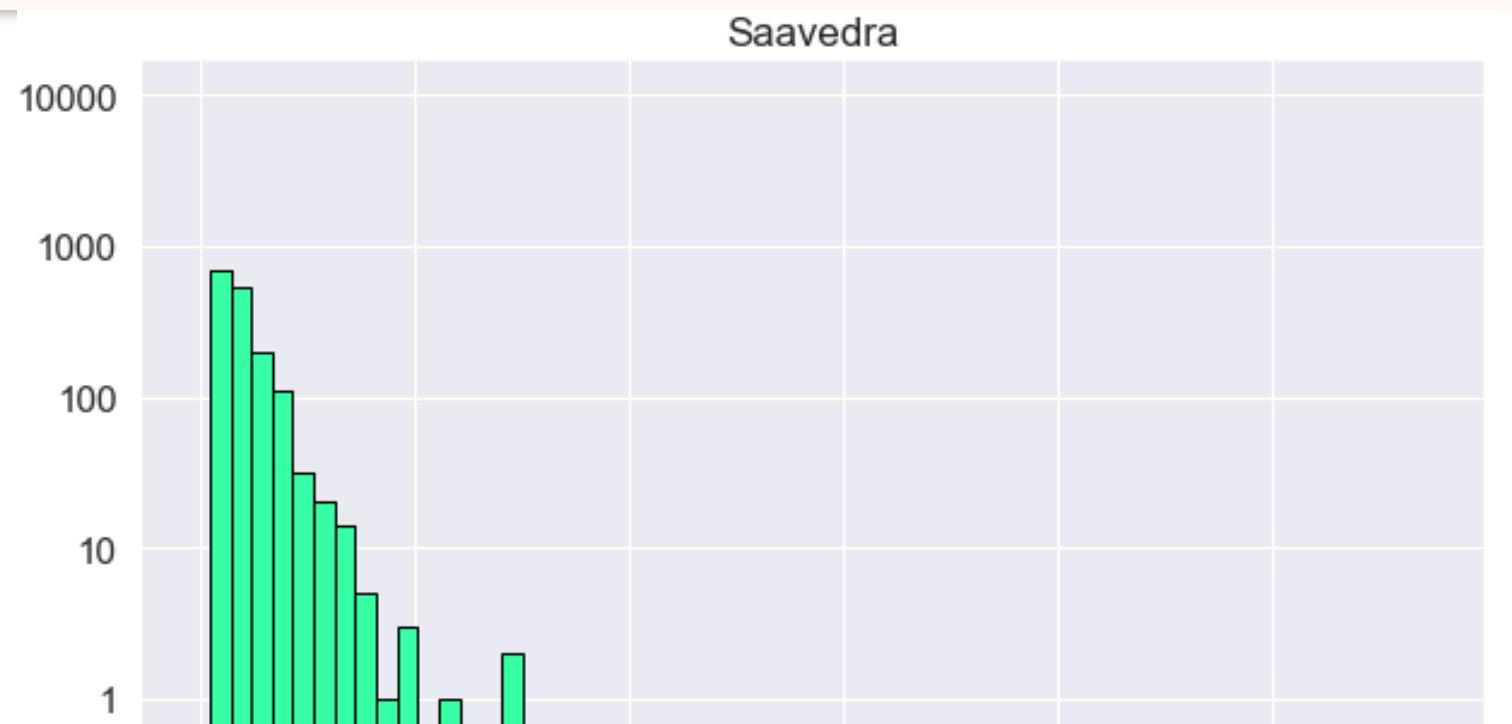
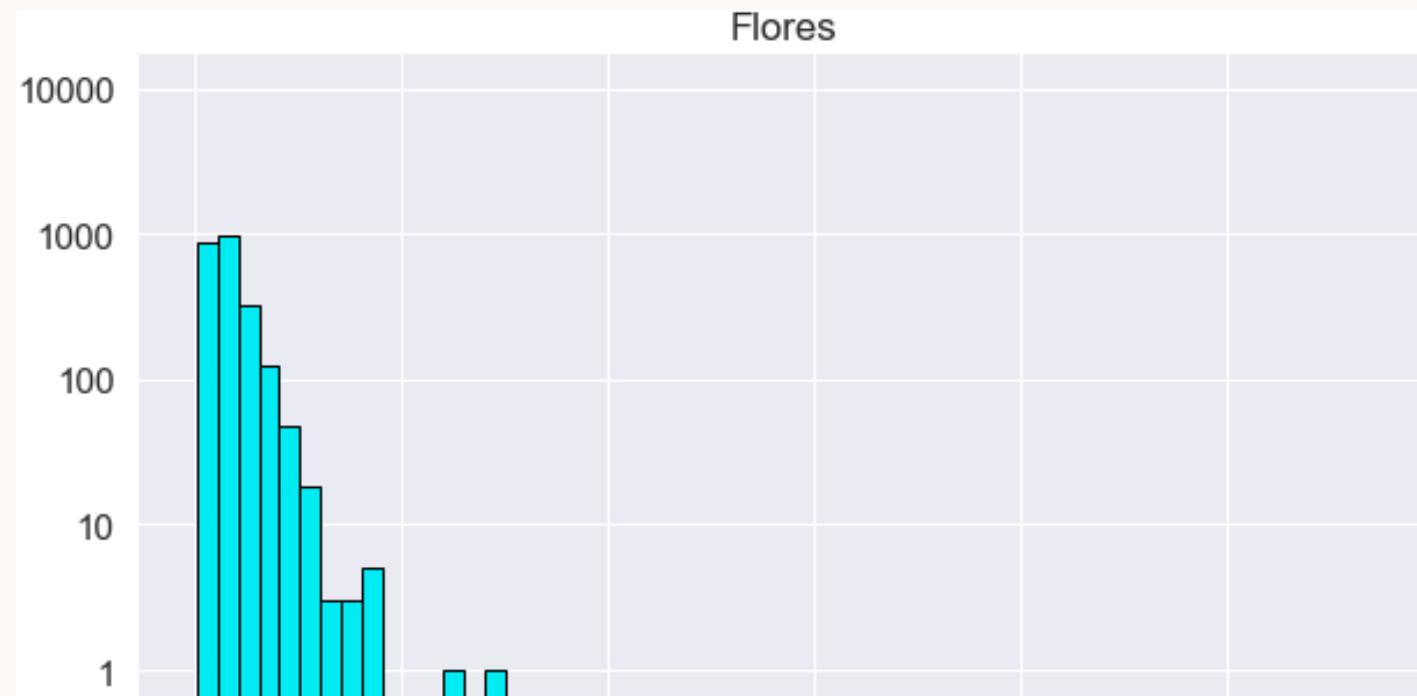
```
0 percentile value is 5000.0
10 percentile value is 78000.0
20 percentile value is 95000.0
30 percentile value is 115000.0
40 percentile value is 132452.0
50 percentile value is 155000.0
60 percentile value is 185000.0
70 percentile value is 229000.0
80 percentile value is 295000.0
90 percentile value is 450000.0
100 percentile value is 5700000.0
```

```
90 percentile value is 450000.0
91 percentile value is 480000.0
92 percentile value is 520000.0
93 percentile value is 560000.0
94 percentile value is 618000.0
95 percentile value is 690000.0
96 percentile value is 790000.0
97 percentile value is 930000.0
98 percentile value is 1200000.0
99 percentile value is 1680000.0
100 percentile value is 5700000.0
```

90% de los datos tiene precio menor a 450 mil dólares
97% de los datos tienen precio menor a 1 millón de dólares

Dataset principal

Observamos algunas distribuciones de los precios de las propiedades por barrio

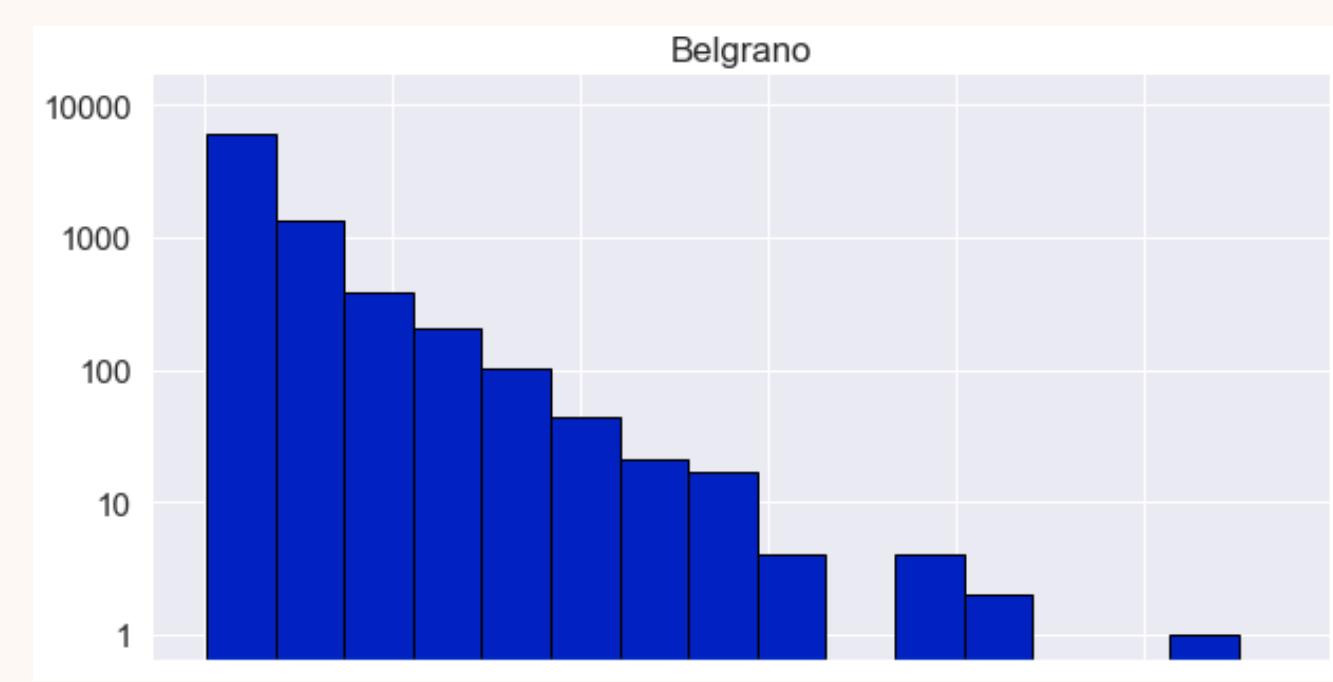
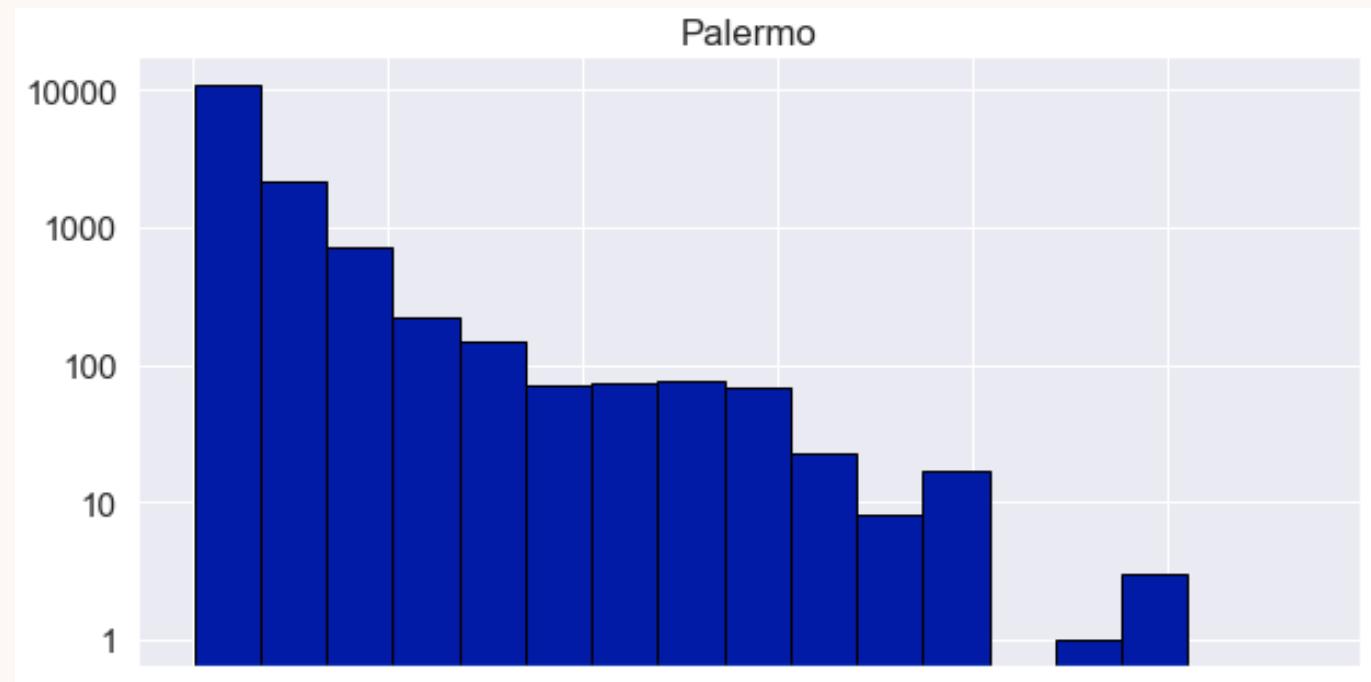
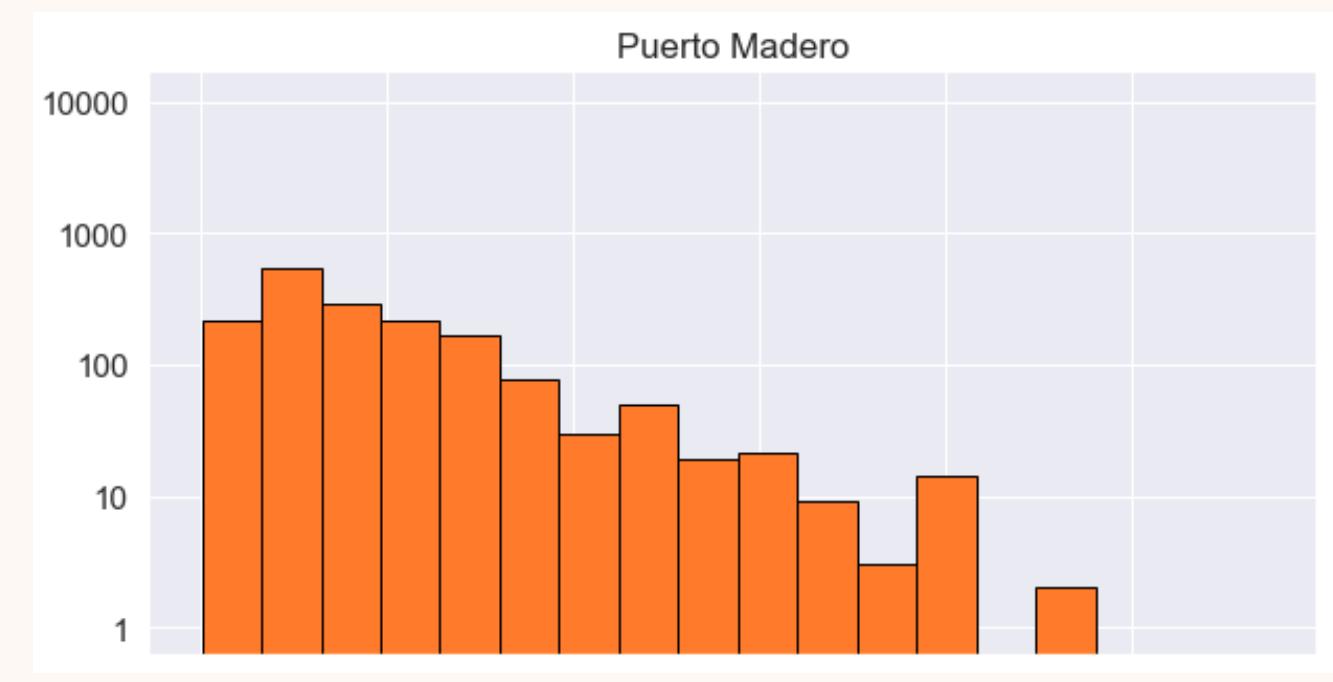
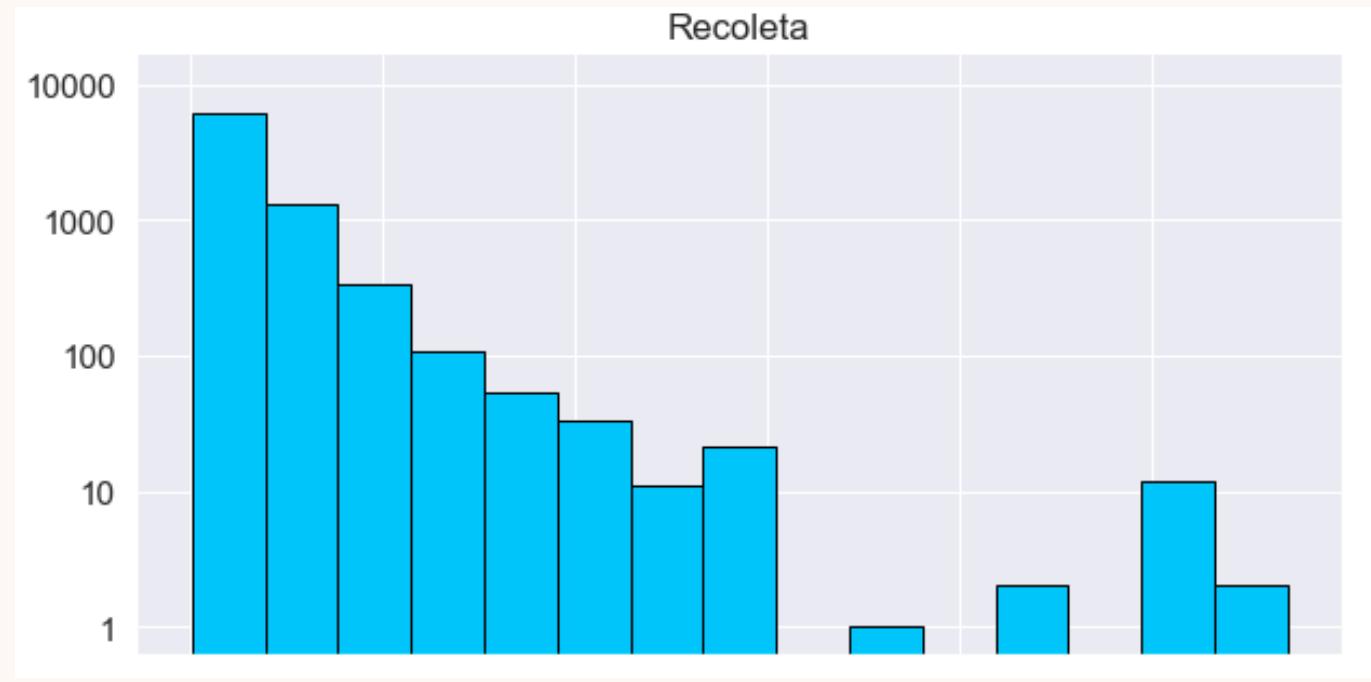


Paso 3: Analizar el dataset principal

3

Dataset principal

Pero hay algunos barrios con más variabilidad



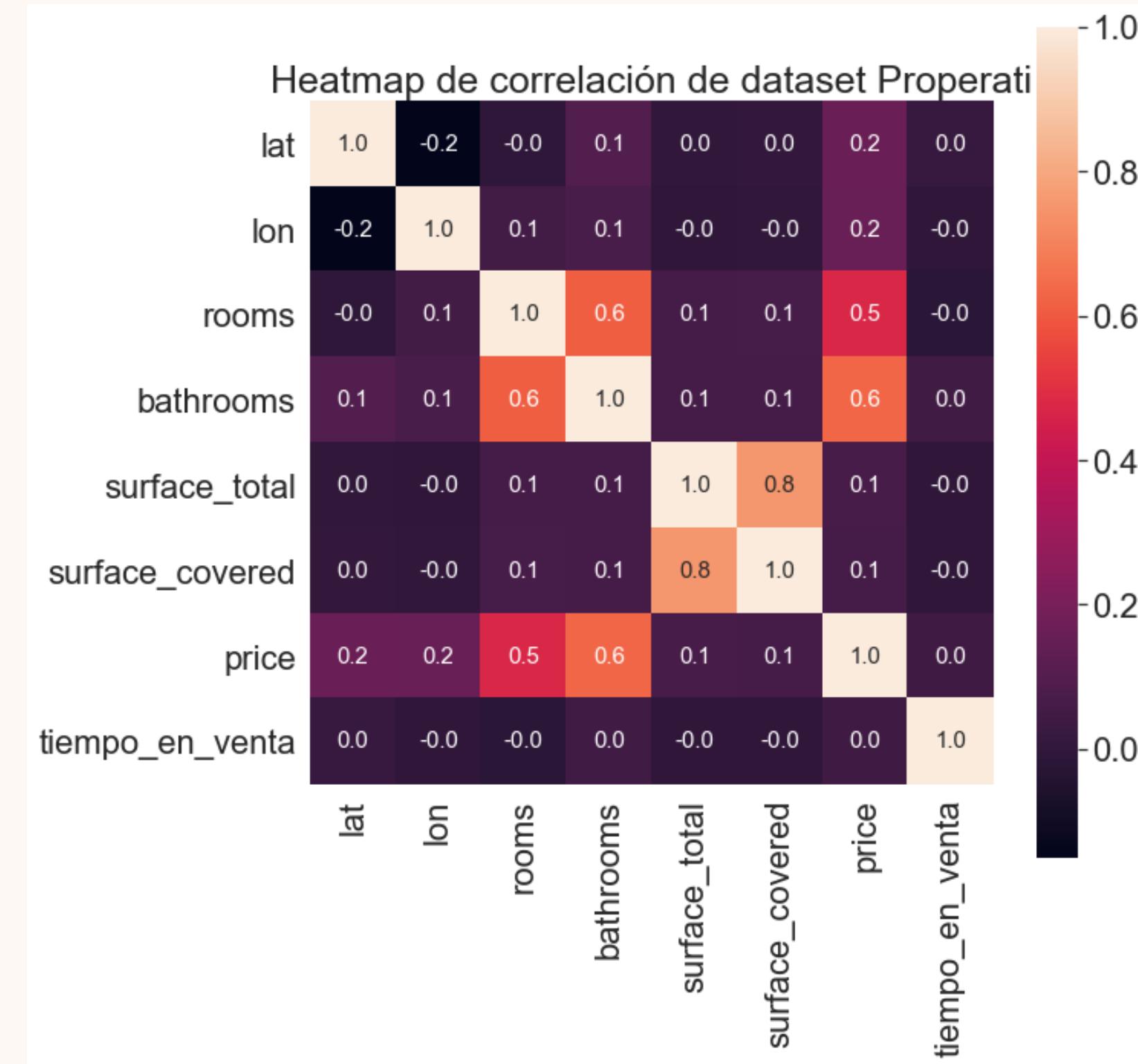
Paso 3: Analizar el dataset principal

3

Dataset principal



Observamos la correlación entre las variables del dataset principal de propiedades



El precio está muy relacionado con los baños y las habitaciones.

Paso 3: Analizar el dataset principal

3

¿Es el dataset principal suficiente?



Se nos ocurrió pensar qué otras variables pueden influir en la predicción de precio de una propiedad:

- Espacios verdes cercanos disponibles,
- Proximidad a los medios de transporte (subte y colectivos),
- Proximidad a barrios marginales,
- Cantidad de escuelas cercanas,
- Nivel de criminalidad de la zona,
- Cantidad de hospitales cercanos, etc

Datasets incluidos

Buscando estos datos, encontramos que por la Ley 27275 de Acceso a la Información Pública (2016), estos datos existen y son abiertos, es decir, todos podemos usarlos.

Espacios verdes

<https://data.buenosaires.gob.ar/dataset/espacios-verdes/resource/juqdkmgo-967-resource>

Metrobus

<https://data.buenosaires.gob.ar/dataset/metrobus/resource/Juqdkmgo-1431222-resource>

Subtes

<https://data.buenosaires.gob.ar/dataset/subte-estaciones/resource/juqdkmgo-1994-resource>

Población por barrio

<https://data.buenosaires.gob.ar/dataset/estructura-demografica/resource/c44be985-8d7f-4aa4-972e-a7f8f0b796dc>

4

Paso 4: Agregar otros datasets

Datasets incluidos

Espacios verdes

Es un dataset proporcionado por el Gobierno de la Ciudad para el año 2019. De allí obtuvimos la columna espacios verdes / población por barrio



Metrobus

Es un dataset proporcionado por el Gobierno de la Ciudad para el año 2019. De allí obtuvimos la columna metrobus_stops_close que corresponde al número de paradas de metrobus más cercanas por cada propiedad en un área de 0.5 km



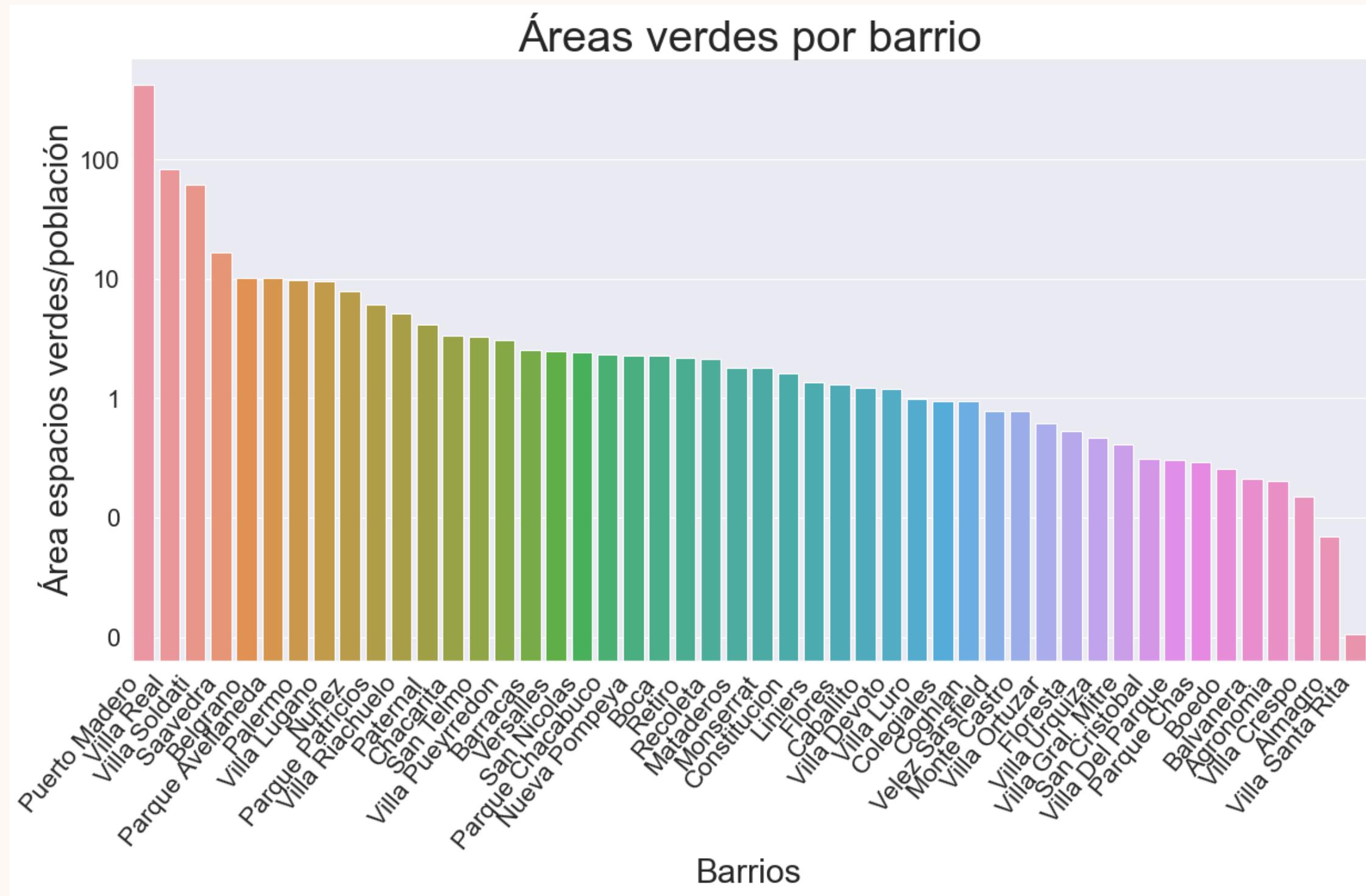
Subtes

Es un dataset proporcionado por el Gobierno de la Ciudad para el año 2019. De ahí obtuvimos la columna subway_stops_close que corresponde al número de paradas de metro más cercanas por cada propiedad en un área de 0,5 km



Espacios verdes

(normalizado por población)



Paso 4: Agregar otros datasets

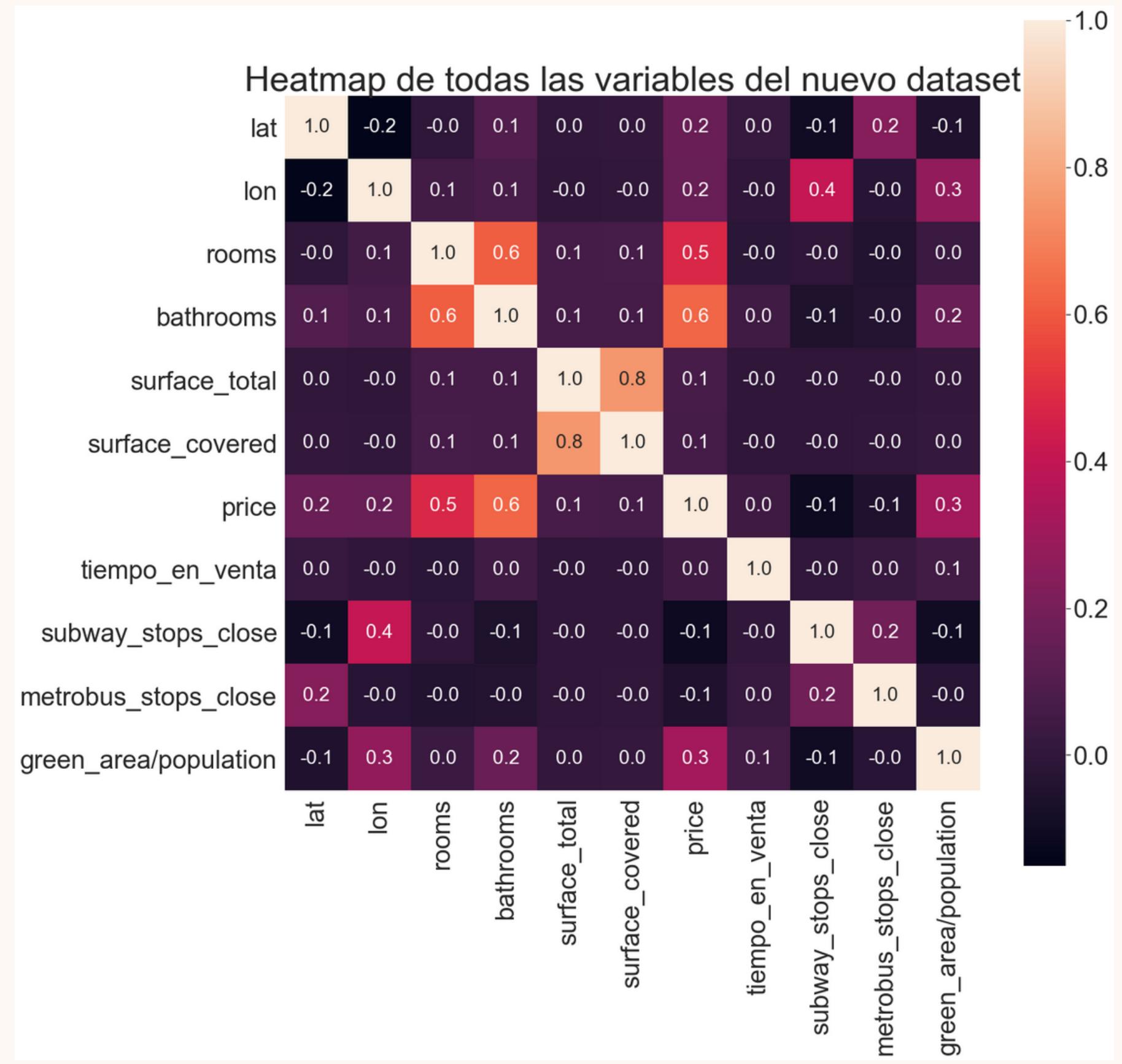
4

Correlación entre todas las variables del dataset

Observamos la correlación

El precio está estrechamente relacionado con las áreas verdes, las estaciones de subte y las paradas de metrobus.

Esto implica que estas nuevas variables influirán en el precio



Paso 4: Agregar otros datasets

4

El dataset queda con las siguientes variables:

```
properties_sale_CABA_new_vars.columns
```

```
Index(['lat', 'lon', 'neighbourhood', 'rooms', 'bathrooms', 'surface_total',
       'surface_covered', 'price', 'property_type', 'tiempo_en_venta',
       'subway_stops_close', 'metrobus_stops_close', 'green_area/population'],
      dtype='object')
```

- latitud
- longitud
- ambientes
- baños
- superficie total
- superficie cubierta
- precio
- paradas de subte cercanas
- paradas de metrobus cercanas
- áreas verdes/población
- tipo de propiedad
- tiempo en venta

Encoding de variables categóricas

Dado que "property_type" es una variable categórica, debemos encodearla (transformarla a variable numérica) para poder aplicar modelos de machine learning.

Para esto, decidimos usar One-Hot Encoding.

The diagram illustrates the process of encoding categorical data. On the left, a table shows a single column labeled 'Color' with five rows containing the values 'Red', 'Red', 'Yellow', 'Green', and 'Yellow'. A large yellow arrow points from this table to the right, indicating the transformation. On the right, a larger table shows the resulting one-hot encoding. The columns are labeled 'Red', 'Yellow', and 'Green'. The rows correspond to the same five entries from the first table. The encoding is binary: a '1' in a row indicates the presence of that color, while a '0' indicates its absence. For example, the first two rows ('Red') both have a '1' in the 'Red' column and '0's in the other two. The third row ('Yellow') has a '1' in the 'Yellow' column and '0's elsewhere. The fourth row ('Green') has a '1' in the 'Green' column and '0's elsewhere. The fifth row ('Yellow') has a '1' in the 'Yellow' column and '0's elsewhere.

Color	Red	Yellow	Green
Red	1	0	0
Red	1	0	0
Yellow	0	1	0
Green	0	0	1
Yellow	0	0	1

También eliminamos la variable "neighbourhood", entonces nos quedan sólo variables numéricas.

Paso 4: Agregar otros datasets

4

Modelos y sus resultados

Aplicamos distintos modelos de machine learning para determinar si las variables agregadas generan una mejora a la predicción

Dataset 1: Datos originales de Properati

Dataset 2: Datos originales de Properati + espacios verdes + subtes + metrobus

Paso 5: Correr los modelos y sus resultados

5

Modelos y sus resultados

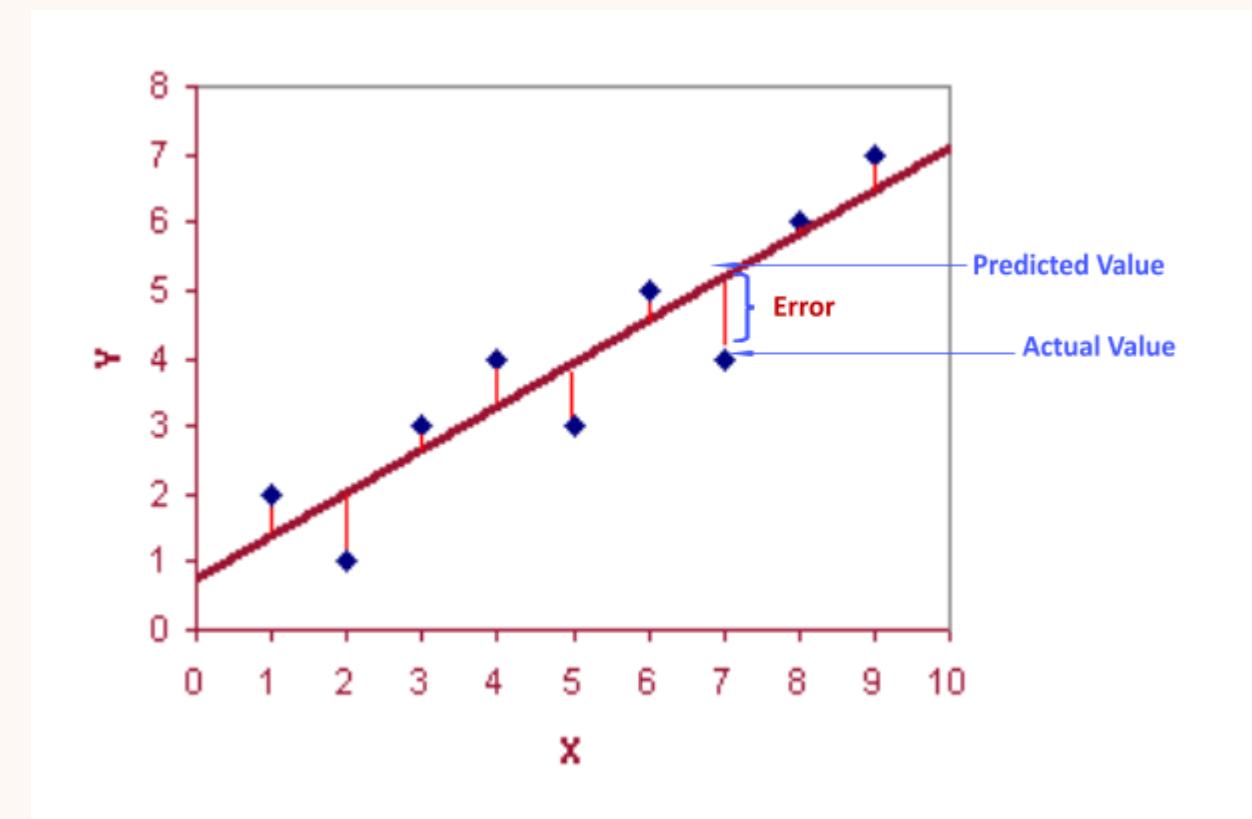
Al analizar los resultados de los modelos a utilizar, optamos por utilizar las siguientes métricas:

- **RMSE**

La raíz del error cuadrático medio (RMSE) indica el ajuste absoluto del modelo a los datos, cuán cerca están los puntos de datos observados de los valores predichos del modelo.

Cuanto más bajo, mejor.

$$RMSE = \sqrt{\frac{1}{M} \sum_{i=1}^M (real_i - estimado_i)^2}$$



Modelos y sus resultados

Al analizar los resultados de los modelos a utilizar, optamos por utilizar las siguientes métricas:

- **R2**

Representa la proporción de la varianza de una variable dependiente que se explica por una o varias variables independientes en un modelo de regresión. Se interpreta como qué tan bien el modelo ajusta los datos. Por ejemplo un R2 de 0.6 significa que el 60% de los datos se ajustan bien con el modelo.

Cuanto más alto, mejor.

$$R^2 = 1 - \frac{SS_{\text{Regression}}}{SS_{\text{Total}}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Modelos y sus resultados

Al analizar los resultados de los modelos a utilizar, optamos por utilizar las siguientes métricas:

- **MAE**

El error medio absoluto(MAE) es similar al RMSE pero en vez de tomar el error cuadrático medio, toma el valor absoluto del error. El RMSE penaliza más los errores grandes, mientras que el MAE penaliza a todos por igual.

Como con el RMSE, cuanto más bajo, mejor.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

test set predicted value actual value

Modelos y sus resultados

Ambos datasets los dividimos en 70% para el training y 30% para el testing



Paso 5: Correr los modelos y sus resultados **5**

Modelos y sus resultados

Para el dataset 1

RMSE: 242146.476

R2 Score: 0.423

MAE: 112005.700

Linear Regression

Para el dataset 2

RMSE: 230719.829

R2 Score: 0.476

MAE: 104811.083

Random Forest Regression

RMSE: 117378.628

R2 Score: 0.864

MAE: 41636.519

RMSE: 113494.838

R2 Score: 0.873

MAE: 39637.497

XGBoost Regression

RMSE: 120729.033

R2 Score: 0.857

MAE: 41911.362

RMSE: 120573.867

R2 Score: 0.857

MAE: 41624.281

Paso 5: Correr los modelos y sus resultados

5

Modelos y sus resultados

Para el dataset 1

RMSE: 242146.476

R2 Score: 0.423

MAE: 112005.700

Para el dataset 2

RMSE: 230719.829

R2 Score: 0.476

MAE: 104811.083

Random Forest Regression

RMSE: 117378.628

R2 Score: 0.864

MAE: 41636.519

RMSE: 113494.838

R2 Score: 0.873

MAE: 39637.497

XGBoost Regression

RMSE: 120729.033

R2 Score: 0.857

MAE: 41911.362

RMSE: 120573.867

R2 Score: 0.857

MAE: 41624.281

Paso 5: Correr los modelos y sus resultados

5

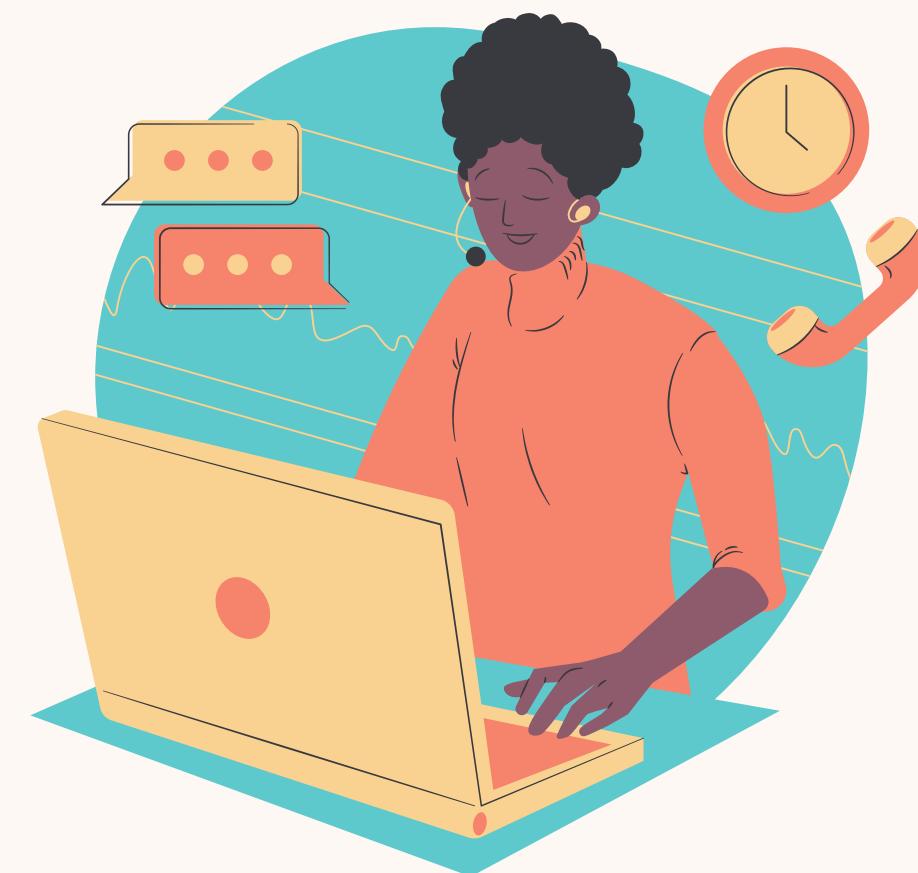
Modelos y sus resultados



El mejor modelo es Random Forest Regressor

Obtuvimos buenos resultados con Random Forest Regressor y XGBoost Regressor pero con Linear Regressor los resultados fueron bastante malos.

Todos los resultados podrían mejorarse aplicando optimización de hiperparámetros (hyperparameter tuning) y técnicas de selección de variables (feature selection).



Paso 5: Correr los modelos y sus resultados

5

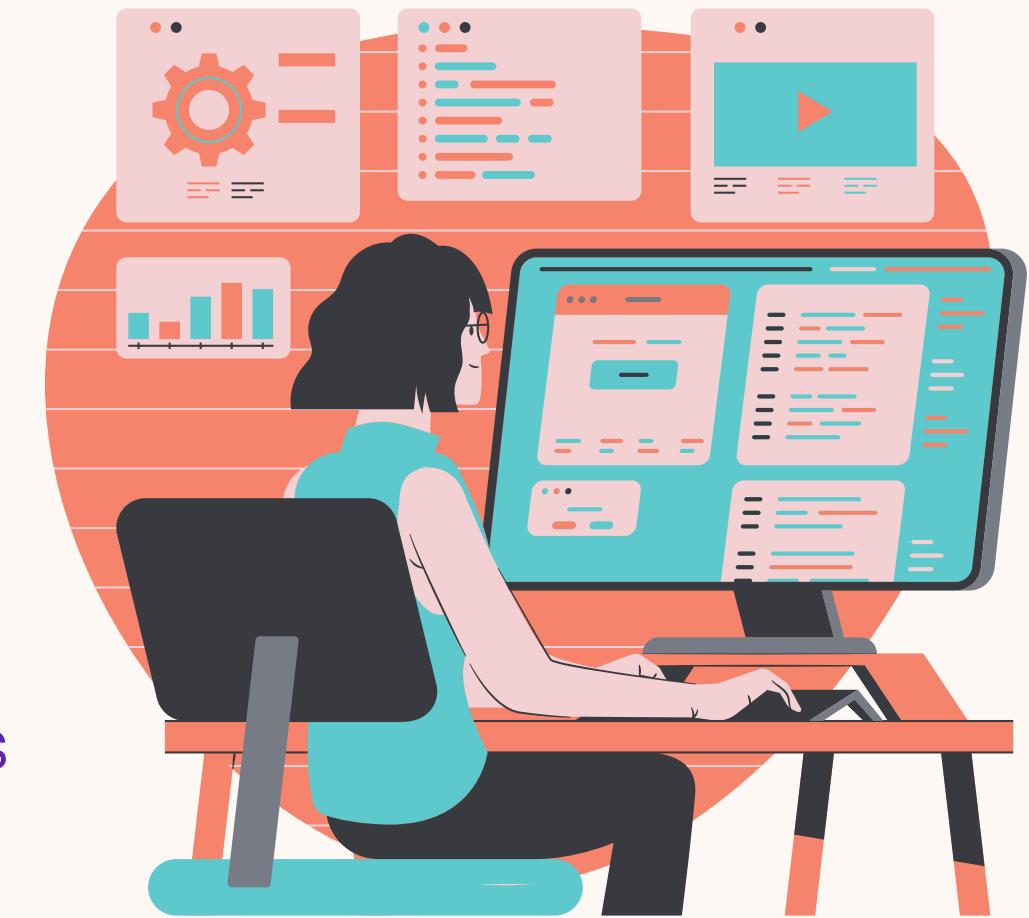
6

Paso 6: Conclusiones

Conclusiones

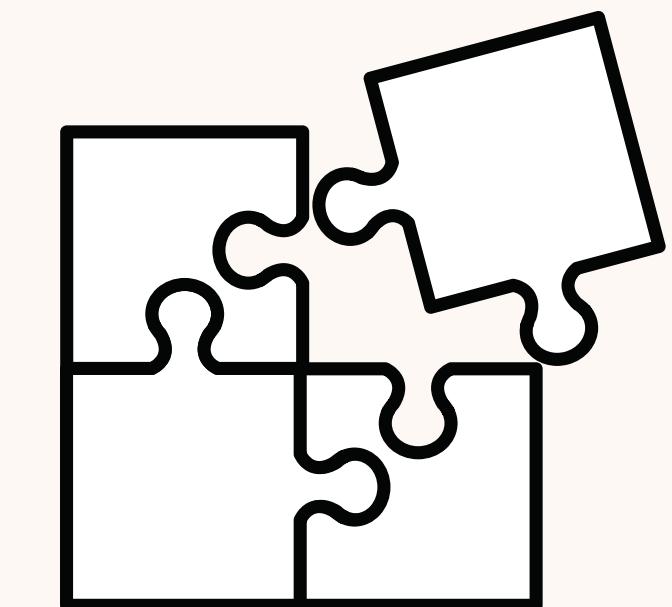
Demostramos que a la hora de decidir el precio de una propiedad, la cantidad de m² de espacios verdes cercanos disponibles normalizado por población, la proximidad a los medios de transporte (subte y colectivos) son factores que mejoran el modelo pero no sustancialmente.

Construimos un conjunto de datos interesante utilizando datos reales y armamos algunos modelos que se pueden usar para predecir el precio de propiedades.



Consideraciones

- La capacidad computacional que tengamos puede llevar a mejores modelos (por ejemplo: la optimización de hiperparámetros toma tiempo para correr)
- Los modelos de machine learning (a diferencia de otros) no están creados con el objetivo de explicar los fenómenos del mundo real, encontrar causalidad y hacer predicciones. Por tanto, hay que ser cuidadosos a la hora de interpretar los resultados y tratar de hacer recomendaciones sobre políticas públicas (Shmueli, 2010)
- La aplicación del aprendizaje automático en la investigación de propiedades aún se encuentra en una etapa temprana (Ho. et al., 2021)



Ho, W. K., Tang, B. S., & Wong, S. w. (2021). Predicting property prices with machine learning algorithms. *Journal of Property Research*, 38(1), 48-70.

Shmueli, G. (2010). To explain or to predict. *Statistical Science*, 25(3), 289–310. <https://doi.org/10.1214/10>

Próximos pasos...

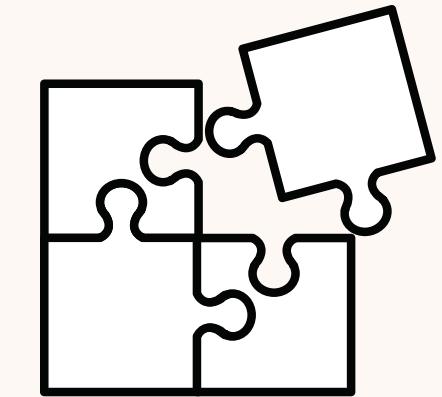
Nos queda pendiente analizar qué otras variables puede mejorar el modelo, para poder realizar mejores predicciones.

Sin embargo:

1) Otros trabajos que también predicen el precio de propiedades con ML muestran que las variables que más correlación presentan con el precio son las que vienen en el dataset original (como cantidad de baños, cuartos, localización, etc.)

2) Los modelos de predicción de precios de propiedades pueden verse influenciados por otras variables inesperadas, como el contexto macroeconómico (Imran et al., 2021).

Sobretodo en Argentina, donde la estabilidad económica cambia constantemente.



Imran, I., Zaman, U., Waqar, M., & Zaman, A. (2021). Using Machine Learning Algorithms for Housing Price Prediction: The Case of Islamabad Housing Data. *Soft Computing and Machine Intelligence*, 1(1), 11-23.

¡Muchas Gracias!



@luciakasman



Lucía Ailén Kasman



Rocío Palacín Roitbarg



github.com/luciakasman/property-data-analysis

