Trabajo Práctico N°1 - Grupo 08

Análisis Exploratorio

Nuestro dataset está compuesto por 460.154 publicaciones (filas) y 20 variables (columnas)

Columnas del dataset:

#	Column	Non-Null Count	Dtype
0	id	460154 non-null	object
1	start_date	460154 non-null	object
2	end_date	460154 non-null	object
3	created_on	460154 non-null	object
4	latitud	419740 non-null	float64
5	longitud	419740 non-null	float64
6	place_l2	460154 non-null	object
7	place_l3	437665 non-null	object
8	place_l4	139020 non-null	object
9	place_l5	2430 non-null	object
10	place_l6	0 non-null	float64
11	operation	460154 non-null	object
12	property_type	460154 non-null	object
13	property_rooms	368498 non-null	float64
14	property_bedrooms	344113 non-null	float64
15	property_surface_total	397813 non-null	float64
16	property_surface_covered	427916 non-null	float64
17	property_price	442153 non-null	float64
18	property_currency	441590 non-null	object
19	property_title	460154 non-null	object

Las variables **cuantitativas** observadas en el dataset son:

- `latitud` (continua): Latitud en la que se encuentra la propiedad.
- `longitud` (continua): Longitud en la que se encuentra la propiedad.
- `property_rooms` (discreta): Cantidad de ambientes con los que cuenta de la propiedad.
- `property_bedrooms` (discreta): Cantidad de dormitorios con los que cuenta la propiedad.
- `property_surface_total` (continua): Superficie total que ocupa la propiedad.
- `property_surface_covered` (continua): Superficie de terreno cubierta con que cuenta la propiedad.
- `property_price` (continua): Precio de la propiedad
- `start_date` (discreta): Fecha de alta del aviso.
- `end_date` (discreta): Fecho de baja del aviso.
- `created_on` (discreta): Fecha de alta de la primera versión del aviso.

En cuanto a variables **cualitativas**, se tiene:

- `id` (nominal): ID de la propiedad.
- `operation` (nominal): Tipo de operación (venta, alquiler, etc.)
- `place_l2` (nominal): Nivel de division administrativa 2, correspondiente a la provincia donde se encuentra la propiedad.
- `place_l3` (nominal): Nivel de division administrativa 3, correspondiente a la ciudad donde se encuentra la propiedad.
- `place_l4` (nominal): Nivel de division administrativa 4, correspondiente al barrio donde se encuentra la propiedad.
- `place_l5` (nominal): Nivel de division administrativa 5. No tiene una equivalencia definida por documentación.
- `place_l6` (nominal): Nivel de division administrativa 6. No tiene una equivalencia definida por documentación.
- `property_type` (nominal): Tipo de propiedad (Casa, Departamento, PH)
- `property_currency` (nominal): Moneda correspondiente al precio publicado.
- `property_title` (nominal): Titulo del anuncio.

Al realizar un filtrado inicial del dataset, conservamos únicamente las propiedades publicadas para la venta, en dólares y que sean Departamentos, Casas o PHs en la CABA.

Preprocesamiento de Datos

Procedemos a eliminar las siguientes variables, dado que las mismas carecen de sentido para el estudio que estamos realizando.

- `id` - No tiene ningún sentido conservar esta columna con los códigos con los que se identificaba a las publicaciones.
- `property_currency` - Dado que hemos filtrado el dataset, dejando solo las publicaciones en dólares, carece de sentido conservar esta columna.
- `operation` - Dado que solo dejamos las operaciones de Venta de propiedades, carece de sentido conservar esta variable.
- `place_l2` - Esta variable contiene la Provincia donde está ubicada la publicación, y como solo hemos dejado las publicaciones ubicadas en CABA, carece de sentido conservarla.
- `place_l4` - Esta variable tiene un 96,13% de valores nulos, carece de sentido conservarla.
- `place_l5` - Esta variable tiene un 100% de valores nulos, carece de sentido conservarla.
- `place_l6` - Esta variable tiene un 100% de valores nulos, carece de sentido conservarla.
- `property_title` - No vemos ninguna utilidad práctica en conservar el título con el que fue publicada la vivienda.
- `created_on` - Esta variable contiene la fecha de alta de la primera versión del aviso, pero al revisar se vio que tiene exactamente los mismos valores que la variable `start_date`, con lo cual se la elimina.

Variables que se conservaron

- `latitud` (continúa): Latitud en la que se encuentra la propiedad.
- `longitud` (continúa): Longitud en la que se encuentra la propiedad.
- `property_rooms` (discreta): Cantidad de ambientes con los que cuenta de la propiedad.
- `property_bedrooms` (discreta): Cantidad de dormitorios con los que cuenta la propiedad.
- `property_surface_total` (continua): Superficie total que ocupa la propiedad.
- `property_surface_covered` (continua): Superficie de terreno cubierta con que cuenta la propiedad.

- `property_price` (continua): Precio de la propiedad
- `start_date` (discreta): Fecha de alta del aviso.
- `end_date` (discreta): Fecha de baja del aviso.
- `place_l3` (nominal): Nivel de división administrativa 3, correspondiente al barrio donde se encuentra la propiedad.
- `property_type` (nominal): Tipo de propiedad (Casa, Departamento, PH)

Se detectaron las siguientes correlaciones entre variables al realizar un heatmap con las variables cuantitativas.

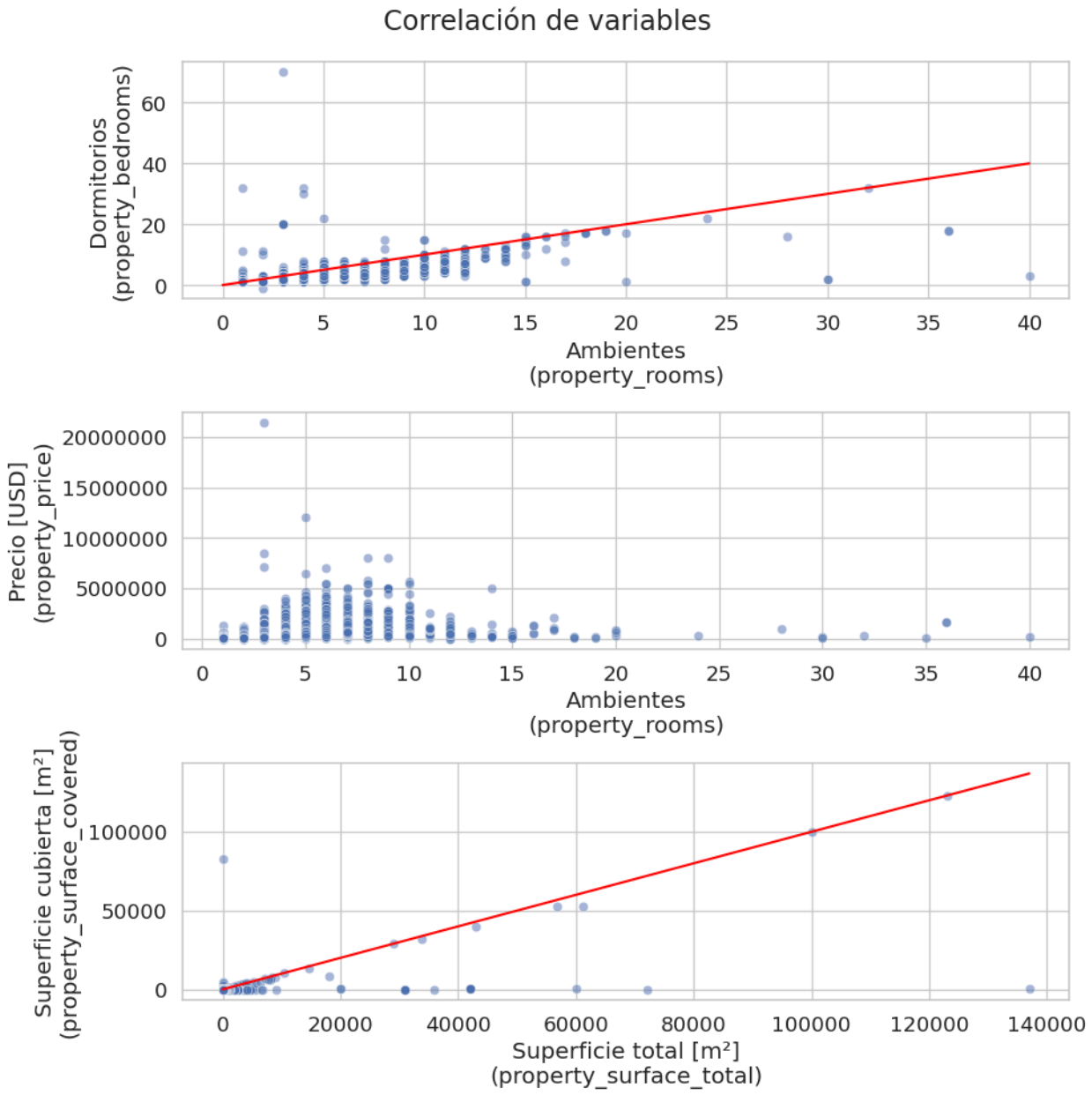
- `property_rooms` y `property_bedrooms (0.85)`
  Podemos observar una correlación positiva, lo cual por supuesto es algo esperable, dado que la cantidad de dormitorios de un inmueble siempre estará acotado superiormente por la cantidad de ambientes del mismo.

- `property_surface_covered` y `property_surface_total (0.6)`
  Al igual que en el caso anterior, observamos una correlación positiva esperable, dado que la superficie total cubierta siempre estará acotada superiormente por la superficie total del inmueble.

- `property_rooms` y `property_price (0.49)`
  Finalmente, observamos una correlación positiva entre la cantidad de ambientes y el precio de la propiedad. Lo cual, era algo esperable.

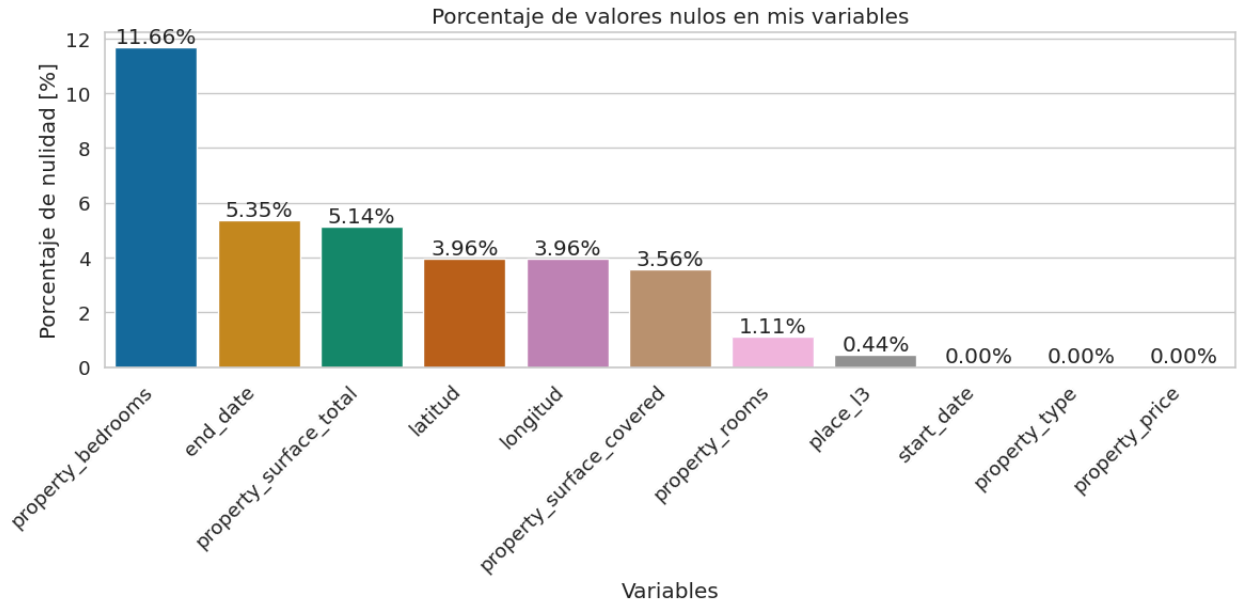
Además, analizamos estas variables mediante gráficas de dispersión, lo cual nos llevó rápidamente a observar que las mismas tenían varios valores atípicos.

- Hay propiedades con más habitaciones que ambientes.
- Hay propiedades con más superficie cubierta que superficie total
- Hay propiedades con un enorme número de ambientes.
- Hay propiedades con una superficie desmesurada.
- Hay propiedades que por su cantidad de ambientes parecen tener un precio desorbitado.



Cómo se manejaron estos valores atípicos se explica más adelante.

2.3.3. Análisis de valores faltantes



1) Se eliminaron las filas que tuvieran valores nulos en las variables Latitud o Longitud. Esto es debido a que posteriormente necesitaremos la ubicación precisa de cada propiedad.
2) Se eliminó la variable end_date dado que a esta altura vimos que carecía de utilidad conservarla.
3) Se eliminaron las propiedades cuya latitud y longitud estuviera fuera de los límites de CABA.
4) Se obtuvo el valor de los barrios que faltaban (variable place_l3) usando los valores de latitud y longitud.

2.3.4. Imputación de datos

1) Imputamos los valores nulos de las columnas `property_bedrooms` (dormitorios) y `property_rooms` (ambientes) mediante un modelo de regresión lineal utilizando IterativeImputer.

   Al modelo le pasamos, además de las columnas mencionadas arriba, las columnas `latitud`, `longitud` y `property_type` para que este pueda utilizarlas como referencia, y así aprovechar mejor la información disponible para imputar los valores faltantes de manera más precisa.

2) Imputamos los valores nulos de las columnas `property_surface_total` (Superficie total) y `property_surface_covered` (Superficie total cubierta) mediante un modelo de regresión lineal utilizando IterativeImputer.

   Al modelo le pasamos, además de las columnas mencionadas arriba, las columnas `latitud`, `longitud`, `property_type` y `property_rooms` para que este pueda utilizarlas como referencia, y así aprovechar mejor la información disponible para imputar los valores faltantes de manera más precisa.

Una vez hecho esto, nos hemos quedado sin valores faltantes.

2.4. Valores atípicos

2.4.1. Analisis Univariado

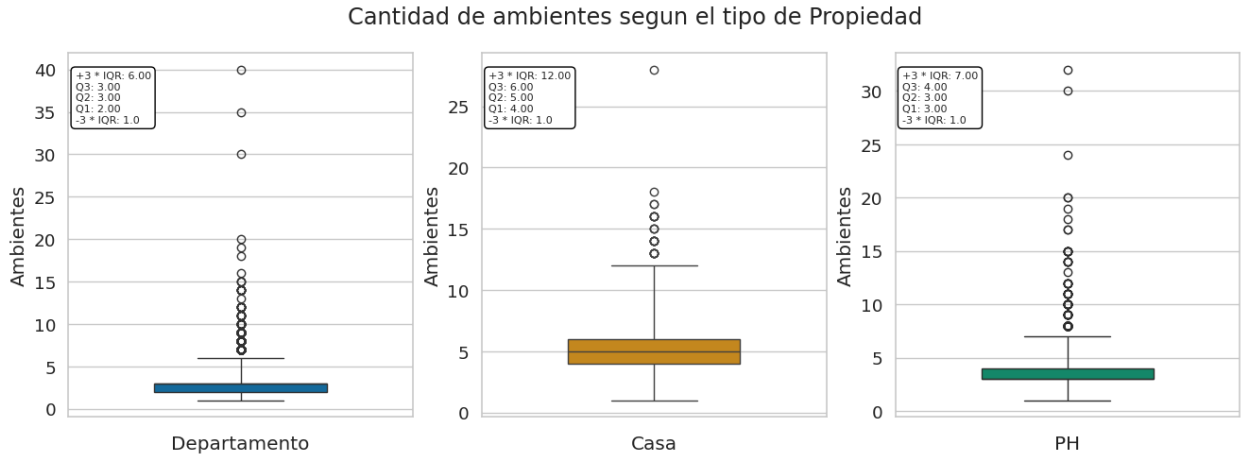
Para la detección de valores atípicos en el dataset se recurrió primero a un análisis univariado de las columnas `property_rooms`, `property_bedrooms`, `property_surface_total`, `property_surface_covered` y `property_price` mediante el uso de boxplots. En este análisis vamos a hacer una distinción según el tipo de propiedad estudiada (Depto, Casa, PH)

Criterios elegidos

En el análisis de los boxplots, consideraremos como outliers a aquellas observaciones que se encuentren a una distancia de 3 veces el rango intercuartílico, en lugar de 1,5 veces (como viene por defecto).

Tomamos esta decisión debido a que observamos que al usar el valor por defecto, perdíamos gran parte de las observaciones que se encontraban dentro de rangos de valores relativamente "comunes" en lo que a propiedades se refiere. De esta forma nos aseguramos que solo se consideran outliers a aquellas observaciones que se alejan en demasía de la media.

2.4.1.1. Variable `property_rooms` (ambientes)



En base a las gráficas arriba desplegada, se obtuvieron las siguientes observaciones:

- En las propiedades de tipo **Departamento** vemos que la gran mayoría de los outliers se ubican por encima de los 6 y hasta los 20 ambientes. Además, vemos que tenemos tres excepciones con 30, 35 y 40 ambientes respectivamente.

- Luego, en las propiedades de tipo **Casa** podemos observar que la gran mayoría de los outliers se ubican por encima de los 12 y por debajo de los 20 ambientes. Ademas, tenemos una propiedad en particular por encima de los 25 ambientes.

- Finalmente, en las propiedades de tipo **PH**, vemos como la mayoría de los outliers oscilan por encima de los 7 y hasta los 15 ambientes. Luego, hay un grupo menor que oscila por encima de los 15 y los 20 ambientes. Y finalmente, algunos casos esporádicos por encima de los 20 ambientes.

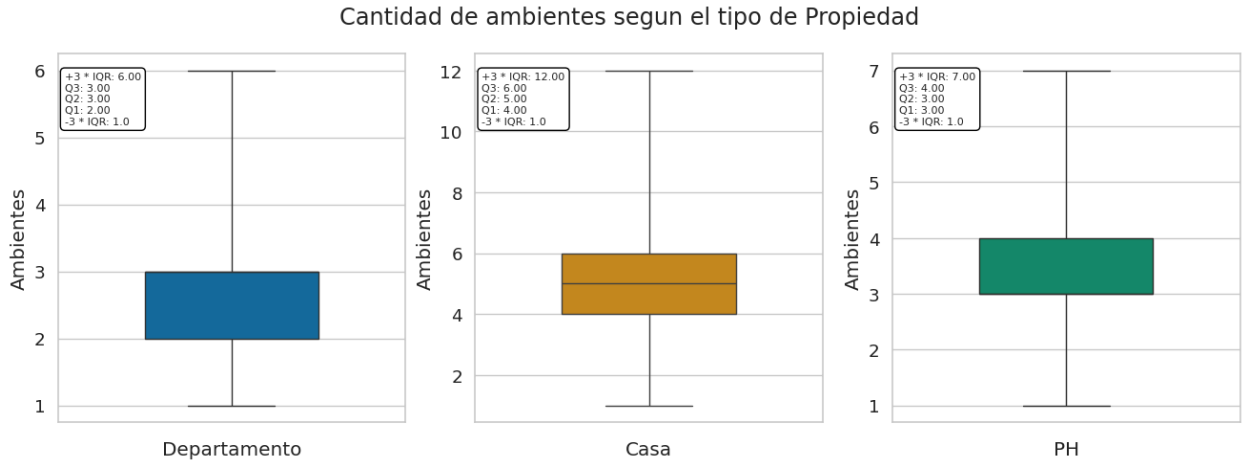
Conclusión:

Concluimos que las propiedades que cuentan con una cantidad de ambientes por encima del valor de su bigote superior es debido a que, o bien ocurrió un error durante la carga de los datos, o bien, se trata de propiedades cuyas características exceden en demasía lo que se consideraría una cantidad de ambientes "normal" para una vivienda.

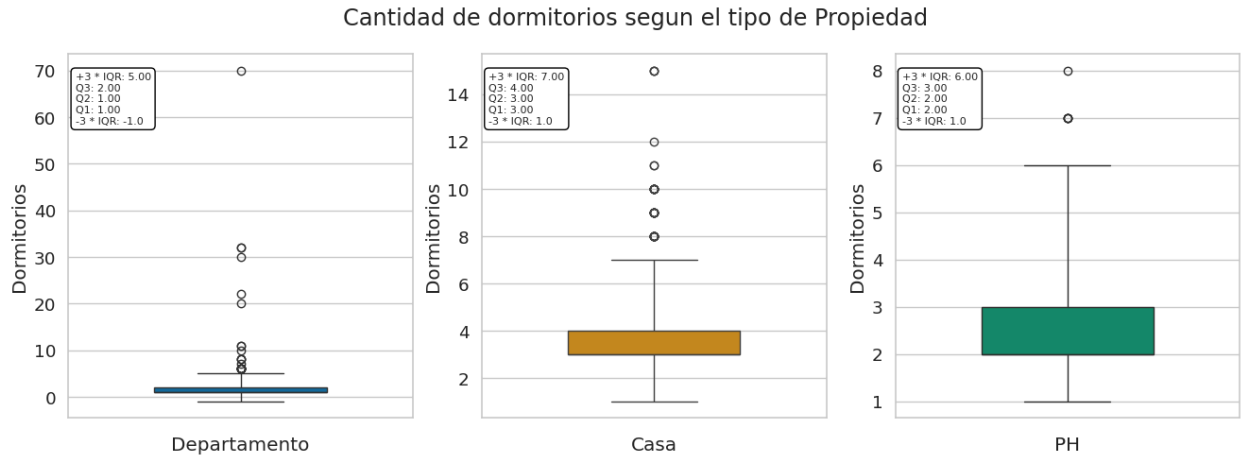
Dado que, consideramos que los valores que estén por encima de esto no son representativos del mercado que se busca estudiar, se decidió eliminar los siguientes registros:

- Departamentos con más de 6 ambientes.
- Casas con más de 12 ambientes.
- PHs con más de 7 ambientes.

Gráfica luego del filtrado de outliers



2.4.1.2. Variable `property_bedrooms` (dormitorios)



Dado que en la sección anterior restringimos el número de ambientes que podían tener los distintos tipos de propiedades, es obvio que estamos ante la presencia de outliers en los tres casos. Pues es imposible que existan viviendas con un número de dormitorios mayor a su número de ambientes.

Además, vemos que hay un caso que llama poderosamente la atención: En los departamentos parece haber propiedades con una cantidad de dormitorios **negativa**.

Al hacer un análisis exhaustivo de este caso, vimos que claramente se trataba de un dato mal cargado, pues era una vivienda de 2 ambientes que decía tener -1 dormitorios. Se corrigió el valor mal cargado.

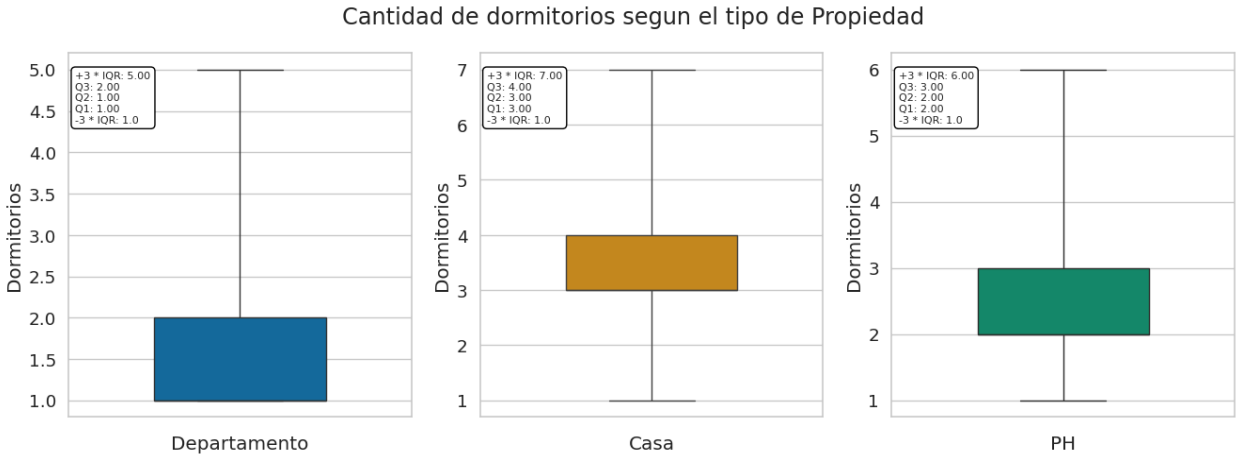
Resuelto ese caso en particular, tenemos que analizar lo siguiente:

En este estudio mediante boxplots puede haber outliers que queden solapados. Por ejemplo: Una casa de 2 ambientes que tenga 3 dormitorios no saldrá como un outlier en el boxplot (esto lo resolveremos más adelante cuando hagamos el análisis multivariado).

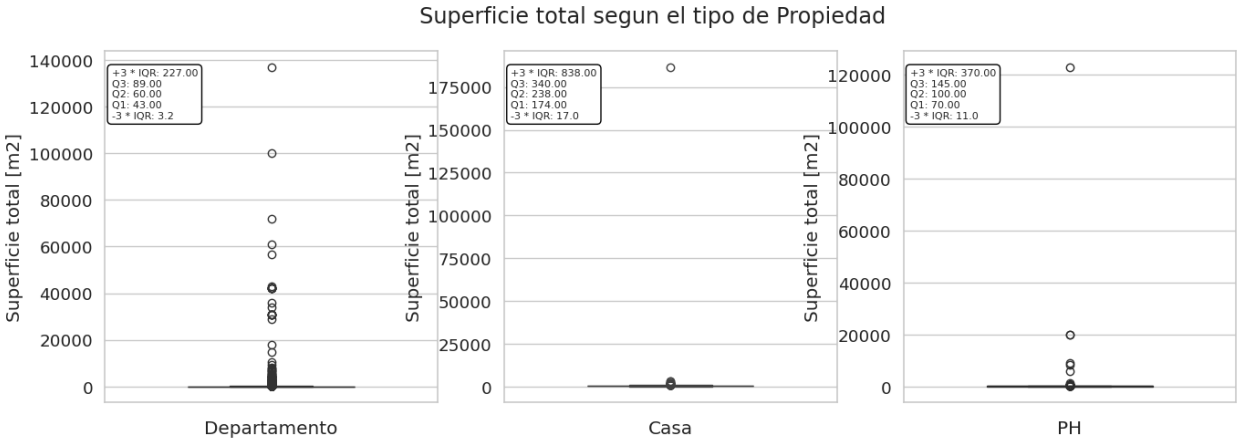
De momento, para poner un límite superior al número de dormitorios, procederemos a eliminar las siguientes propiedades:

- Departamentos que cuenten con más de 5 dormitorios.
- Casas que cuenten con más de 7 dormitorios.
- PHs que cuenten con más de 6 dormitorios.

Gráfica luego del filtrado



2.4.1.3. Variable `property_surface_total` (superficie total)

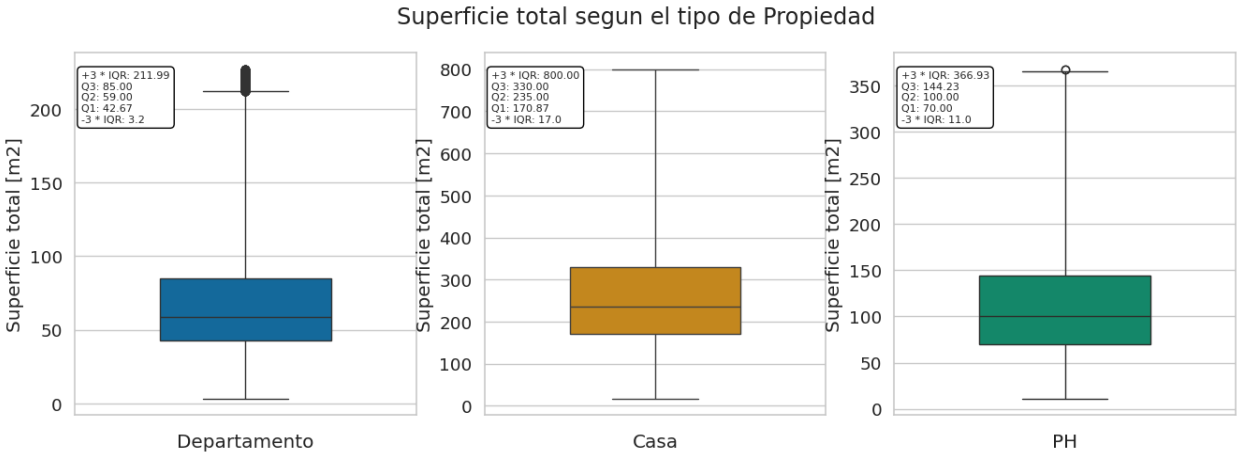


Como podemos observar en las gráficas, tenemos un gran número de observaciones cuya superficie no se condice con lo que uno esperaría encontrar en la realidad. Sobre todo en el caso de los departamentos, donde hay observaciones con una superficie de 100.000 m2 (aproximadamente 14 canchas de fútbol)

De momento estableceremos como cota superior para la superficie de nuestras propiedades el valor de sus bigotes superiores.

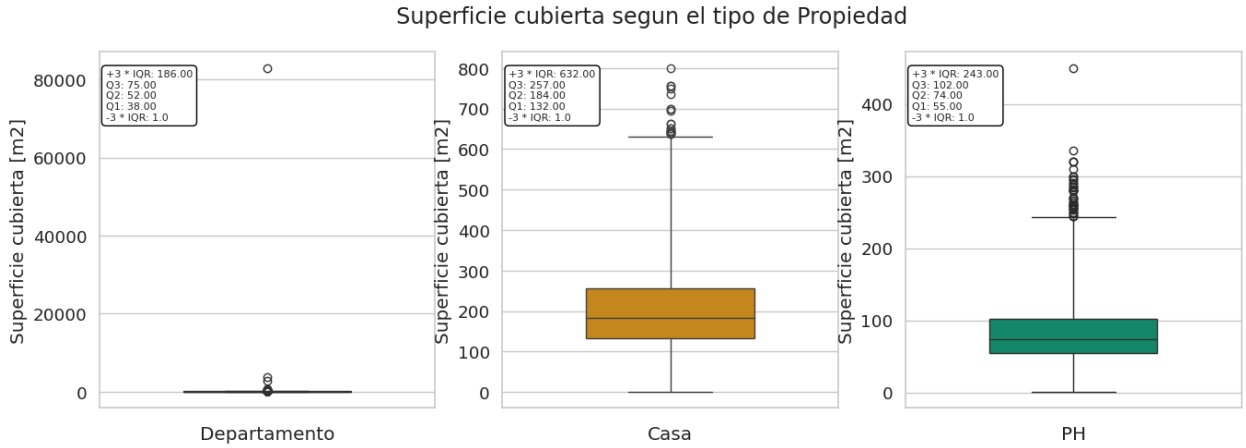
- Departamentos ▸ Máximo 227 m2
- Casas ▸ Máximo 834 m2
- PHs ▸ Máximo 370 m2

Gráfica luego del filtrado



Más adelante, en el análisis multivariado, haremos un hilado más fino. Dado que puede haber outliers que queden solapados en este análisis mediante boxplots (por ejemplo, un monoambiente con 200 m2)

2.4.1.4. Variable `property_surface_covered` (superficie cubierta)

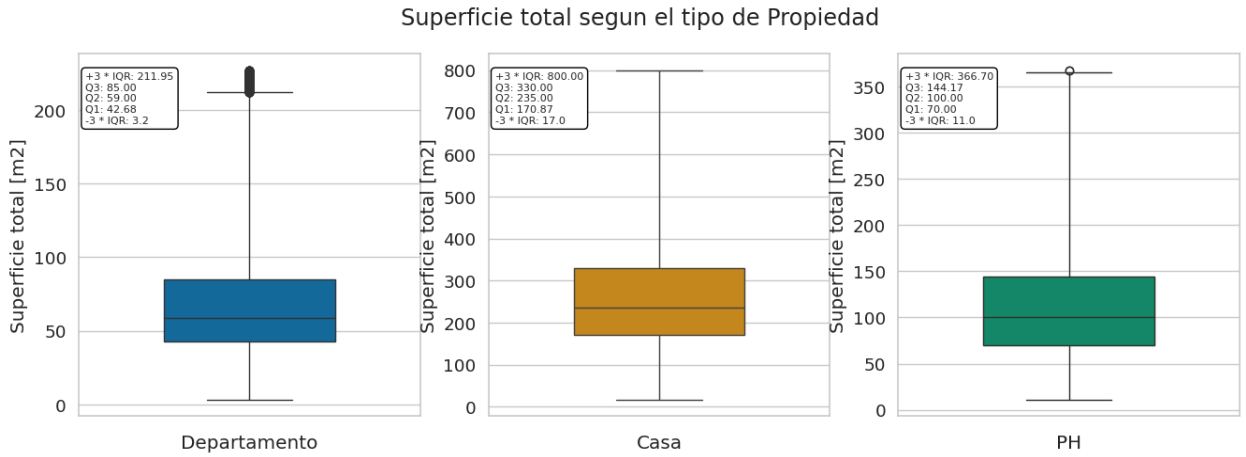


Como podemos observar en las gráficas, ocurre algo muy similar a lo que veíamos con la variable `property_surface_total`.

De momento estableceremos como cota superior para la superficie cubierta los mismos valores que habíamos asignado a la variable `property_surface_total`. Dado que me está dando límites muy bajos el boxplot y no quiero perder tanta data de golpe.

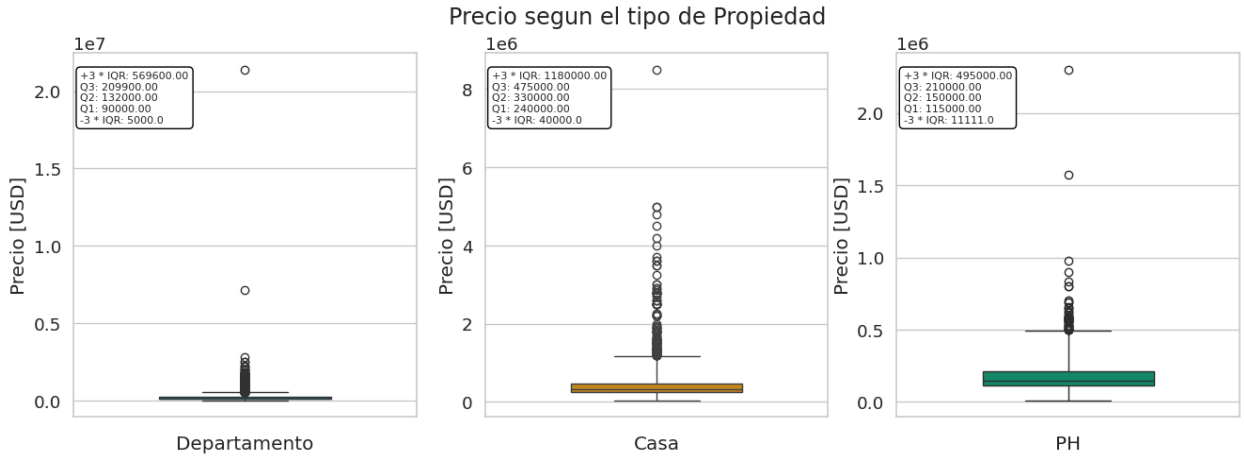
- Departamentos ▸ Máximo 246 m2
- Casas ▸ Maximo 841 m2
- PHs ▸ Maximo 367 m2

Gráfica luego del filtrado



Por ahora lo vamos a dejar así y vamos a hacer un ajuste más fino en análisis multivariado.

2.4.1.5. Variable `property_price` (Precio)



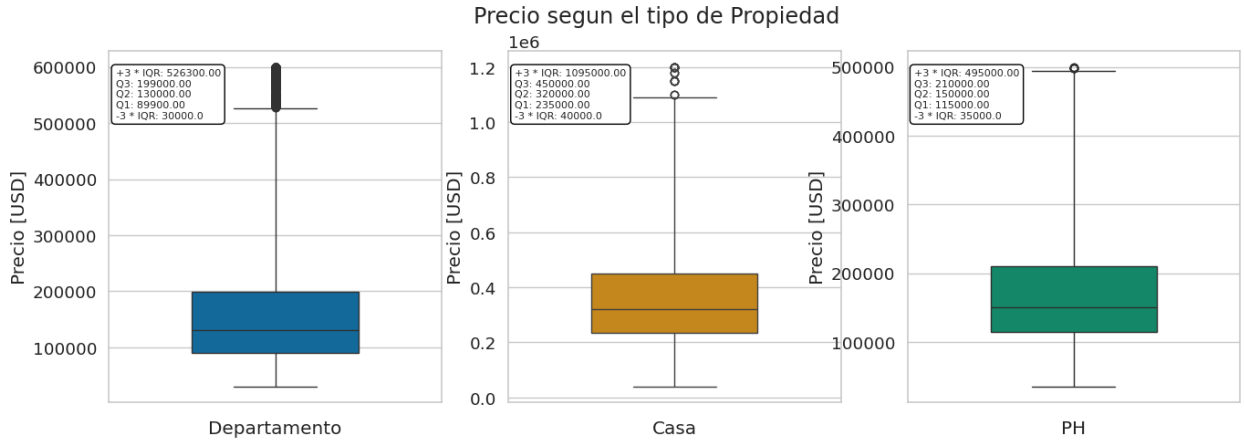
En esta variable es en donde, probablemente, sea más inutil hacer un análisis univariado. Dado que el precio de una propiedad varía sustancialmente en base a parámetros como la ubicación, la superficie y la cantidad de ambientes de la misma.

En primera instancia solo vamos a conservar:

* Departamentos ▸ Precio por debajo de 600 K
* Casas ▸ Precio por debajo de 1,2 M
* PHs ▸ Precio por debajo de 500 K

Además, vamos a poner un límite inferior de 30 K a todas las propiedades, pues consideramos que es ilógico que haya propiedades en venta por un precio tan bajo.

Gráfica luego del filtrado



2.4.2. Análisis Multivariado

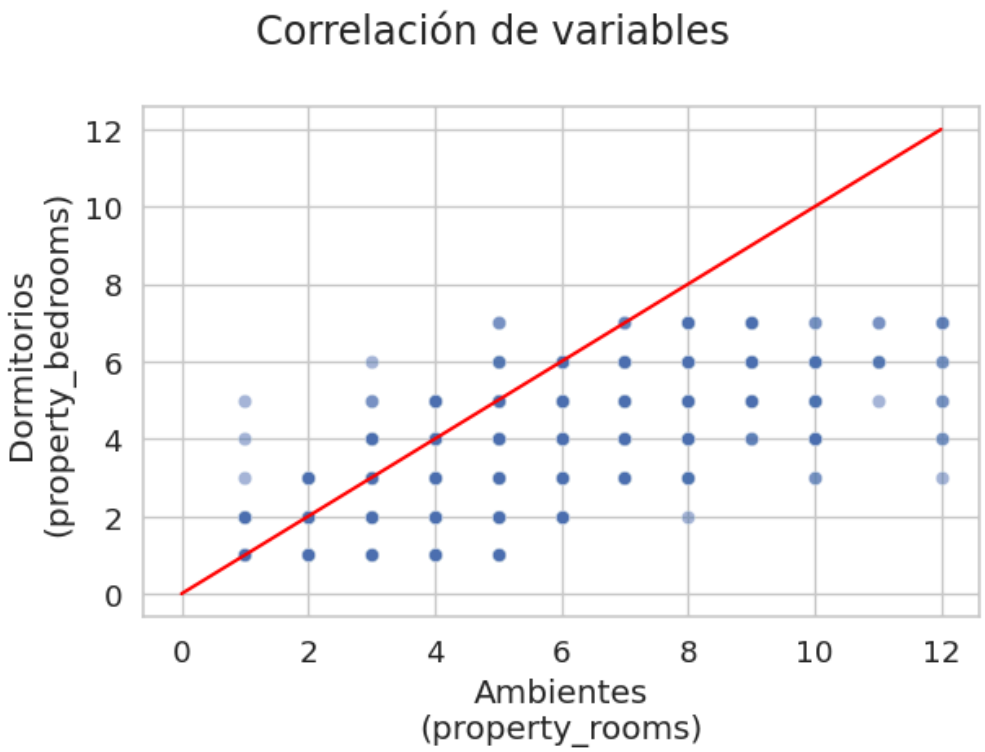
2.4.2.1. `property_rooms` (ambientes) vs `property_bedrooms` (dormitorios)

Dado que, como dijimos en la sección 2.4.1.2, pueden haber quedado outliers solapados en mis datos. Por ejemplo: Una casa de 2 ambientes que tenga 3 dormitorios. Haremos lo siguiente:

- Las propiedades de 1 ambiente, deberán tener 1 dormitorio.
- Las propiedades de n ambientes, podrán tener como máximo n-1 dormitorios (n > 1)

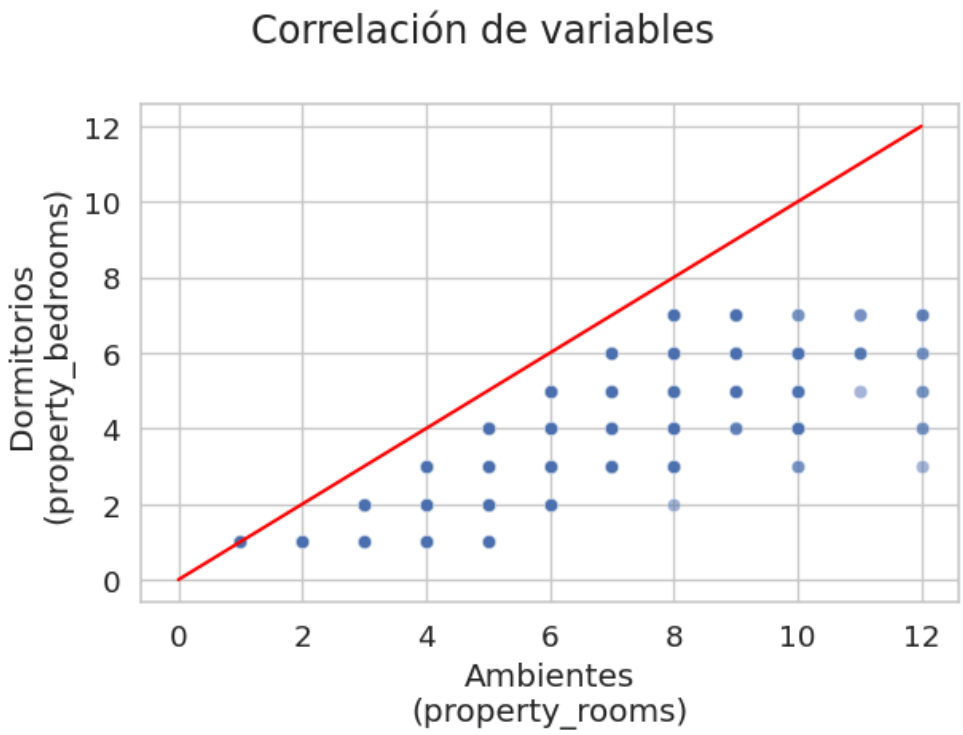
Considerando que, si bien el dataframe indica `property_bedrooms` (es decir, cantidad de dormitorios) en la venta de propiedades lo que se suele relevar es el número de **habitaciones** con que cuenta la propiedad sobre el número de ambientes.

Es decir:
- En una casa de 2 ambientes, tiene 1 habitación.
- En una casa de 3 ambientes, tiene 2 habitaciones.
- En una casa de 4 ambientes, tiene 3 habitaciones.
- etc.



Como podemos observar, hay propiedades en las cuales tenemos más dormitorios que ambientes. Vamos a corregir eso.

Gráfica luego del filtrado



2.4.2.2. `property_rooms` (ambientes) vs `property_surface_total` (superficie)

En la sección 2.4.1.3 al realizar el análisis univariado de la variable property_surface_total lo que hicimos fue establecer una cota superior para la superficie máxima que podían tener los distintos tipos de propiedades.

Esto en el caso de las Casas y los PHs no es tan problemático, pues una casa puede tener tranquilamente 2 ambientes, pero 100 m2 de patio.

El problema de este filtrado surge con los Departamentos donde se puede dar, por ejemplo, el caso de tener un monoambiente que con 200 m2 de superficie.

Para resolver esto recurriremos a un dataset del gobierno de la CABA, en el cual hay un relevamiento de la superficie promedio (en metros cuadrados) en función de la cantidad de ambientes de departamentos publicados entre 2010 y 2024.

Dado que este dataset solo contiene data de deptos de 1 a 3 ambientes, tendremos que estimar la superficie de los deptos con un número de ambientes mayor por otro medio.

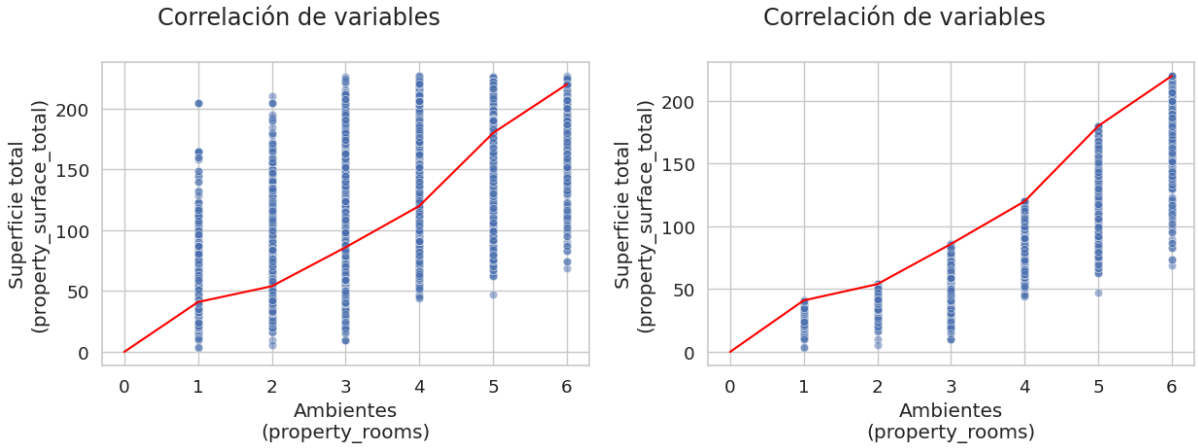
Como regla práctica, cada tipo de Departamento podrá tener como máximo un 25% de superficie por encima del valor que estimo nuestro modelo. (redondeando valores para mayor practicidad)

Para departamentos de 4 a 6 ambientes decidimos tomar los siguientes valores máximos:

* Para 4 ambientes, la superficie máxima será 120 m2
* Para 5 ambientes, la superficie máxima será 180 m2
* Para 6 ambientes, la superficie máxima será 220 m2

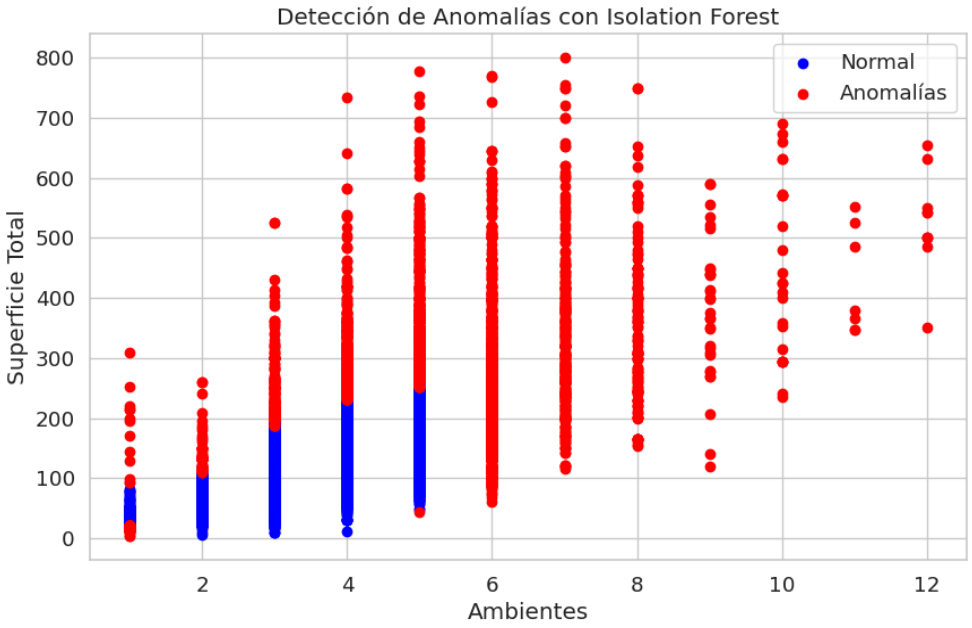
Estos valores se estimaron en base a observaciones realizadas sobre las viviendas publicadas en los sitios Zonaprop y Argenprop.

Gráfica antes de filtrar y luego de filtrar



La línea roja es el límite de superficie que fijamos en función del número de ambientes de los departamentos.
Si bien esto ayudó a corregir los máximos, aún falta purgar bastante los mínimos. Pues como podemos ver, tengo deptos de 6 ambientes con una superficie de 50 m2

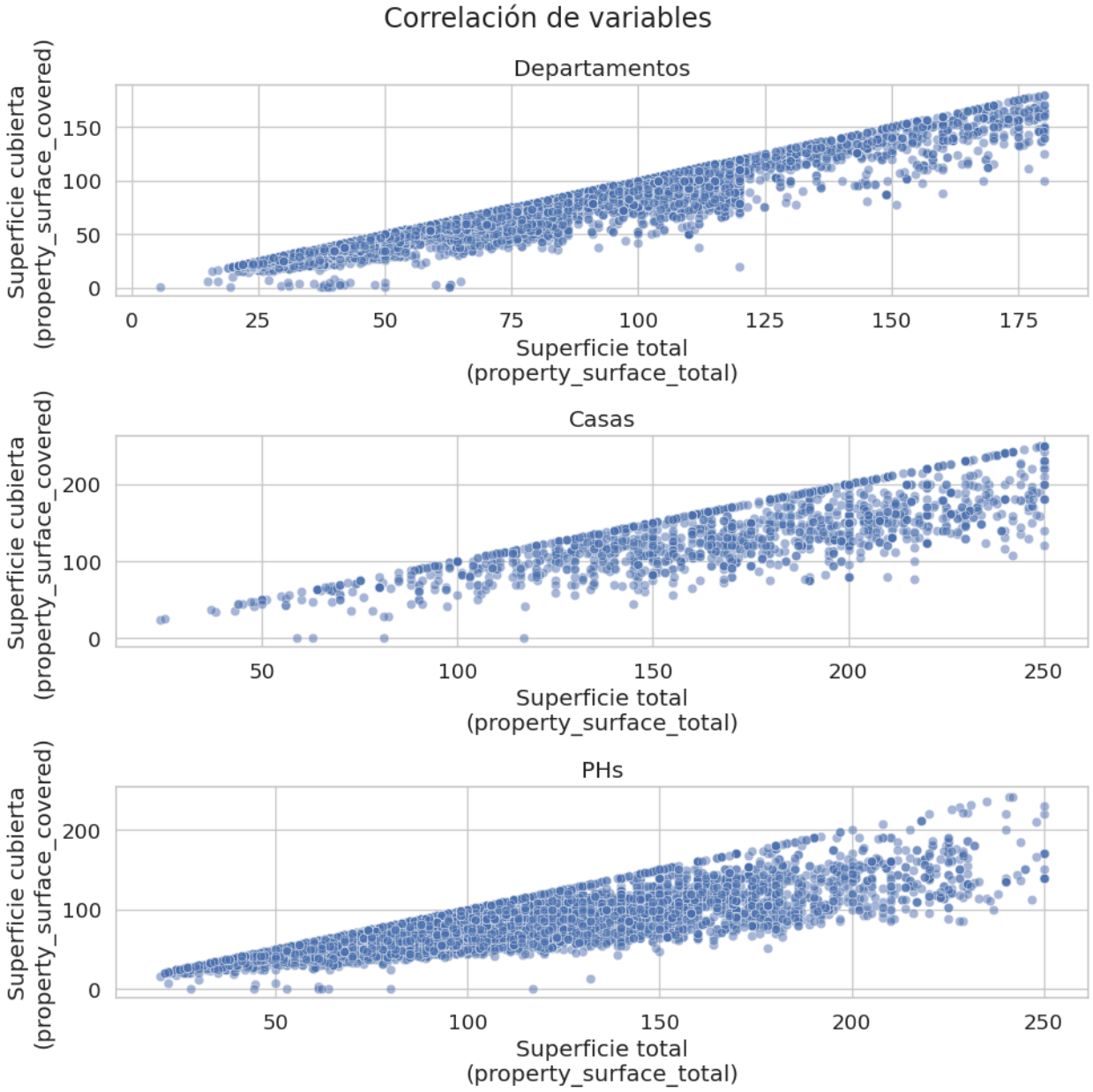
Para un filtrado más fino recurrimos a un Insolation Forest



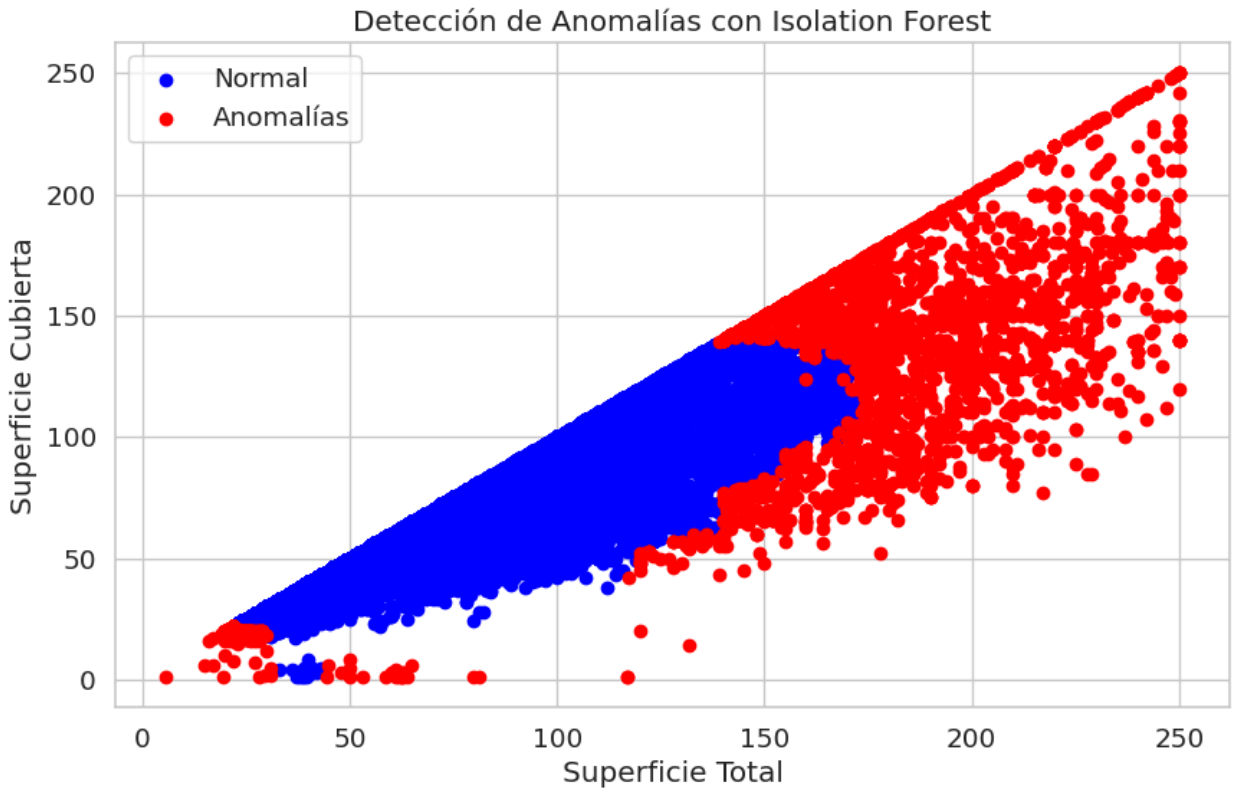
2.4.2.3. `property_surface_total` (superficie total) vs `property_surface_covered` (superficie cubierta)

Primero que nada graficamos una variable versus la otra y eliminamos todas las propiedades cuya superficie cubierta sea mayor a su superficie total.

Gráfica luego del filtrado inicial.



Luego, para un filtrado más fino, recurrimos a un Insolation Forest



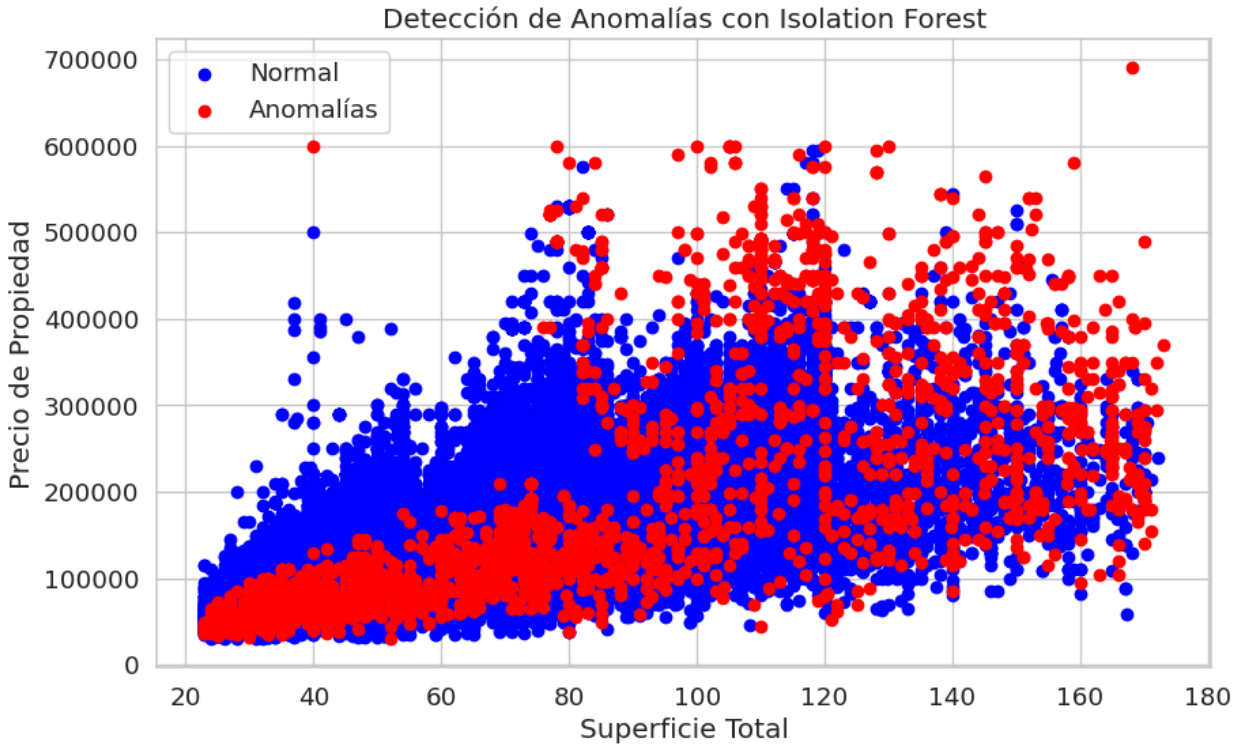
2.4.2.4. `property_surface_total` (Superficie total) vs `property_price` (Precio)

Vamos a analizar el precio de las propiedades comparándolo contra la superficie de las mismas para buscar anomalías.

Pero, hay un problema que salta a la vista en cuanto pensamos en relacionar estas dos variables ▸ El precio por metro cuadrado puede llegar a variar considerablemente en función del barrio en el cual nos encontremos.

Para intentar compensar esto, vamos a tener en cuenta también el barrio en el cual esté ubicada la propiedad.

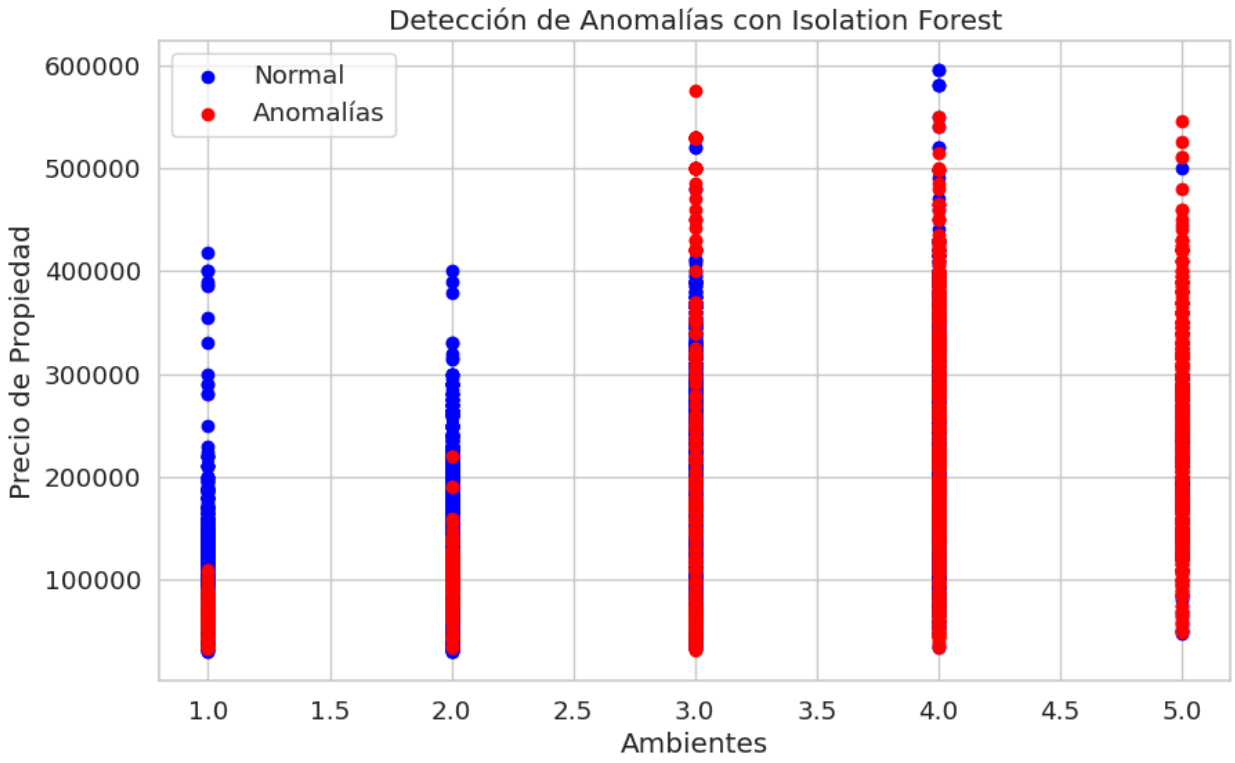
Gráfica luego del filtrado



2.4.2.5. `property_rooms`(ambientes) vs `property_price` (precio)

Similar a lo que hicimos en la sección anterior, ahora vamos a analizar el precio de las propiedades comparándolo contra la cantidad de ambientes para buscar anomalías.

Al igual que antes, también vamos a tener en cuenta el barrio en el cual se encuentra la propiedad ubicada, dado que no cuesta lo mismo un 2 ambientes en Recoleta que uno en Floresta.



Visualizaciones

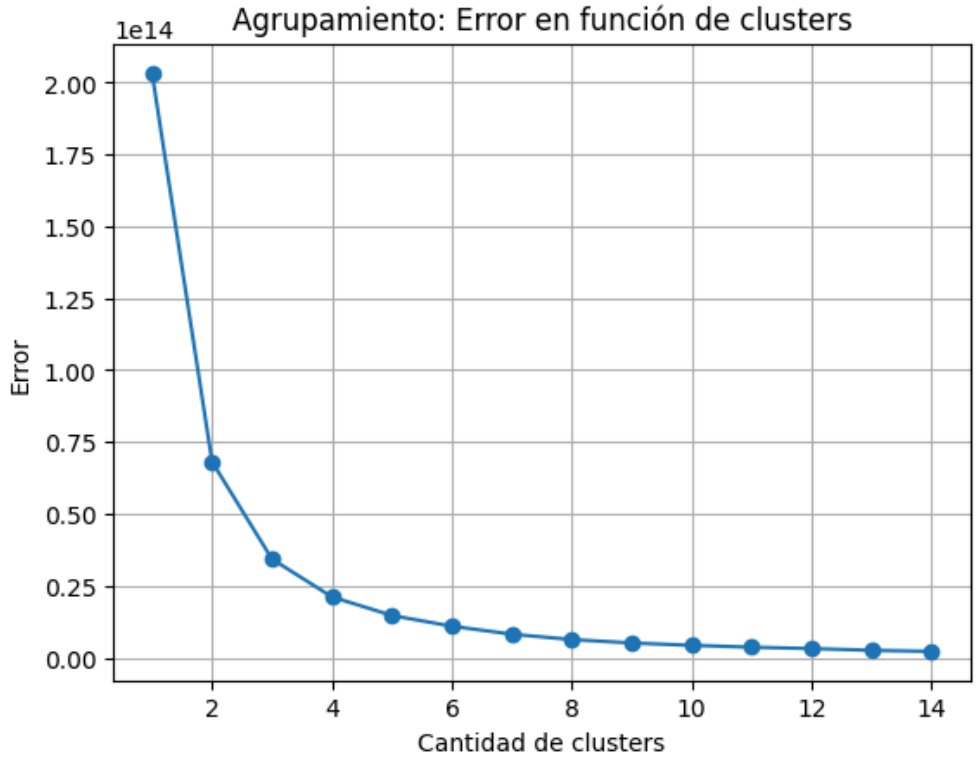
Esto se realizó en la sección de análisis y procesamiento.

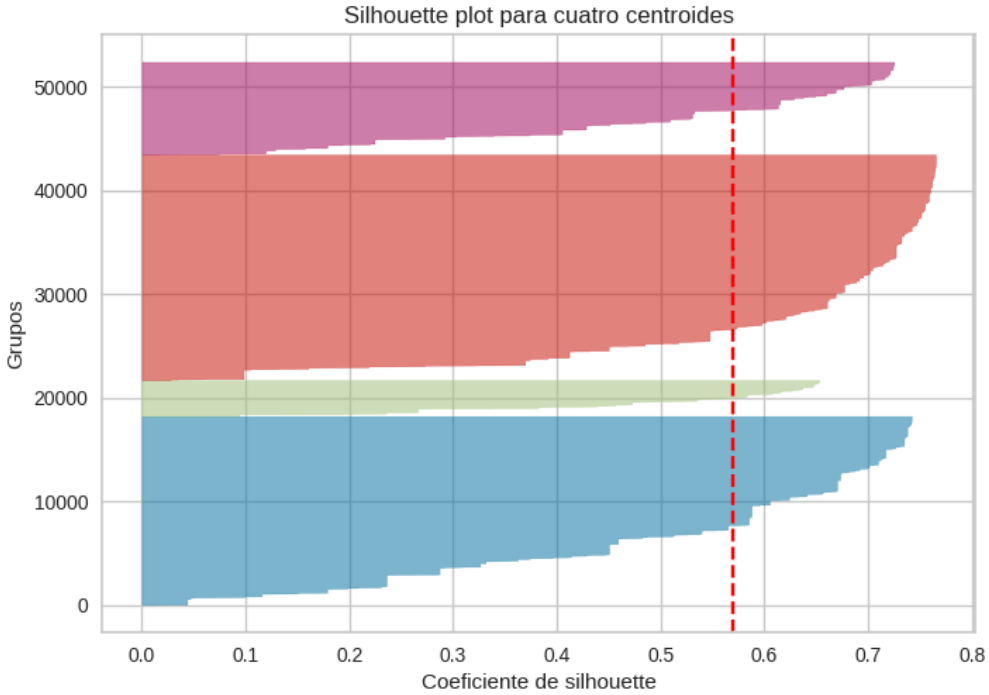
Clustering

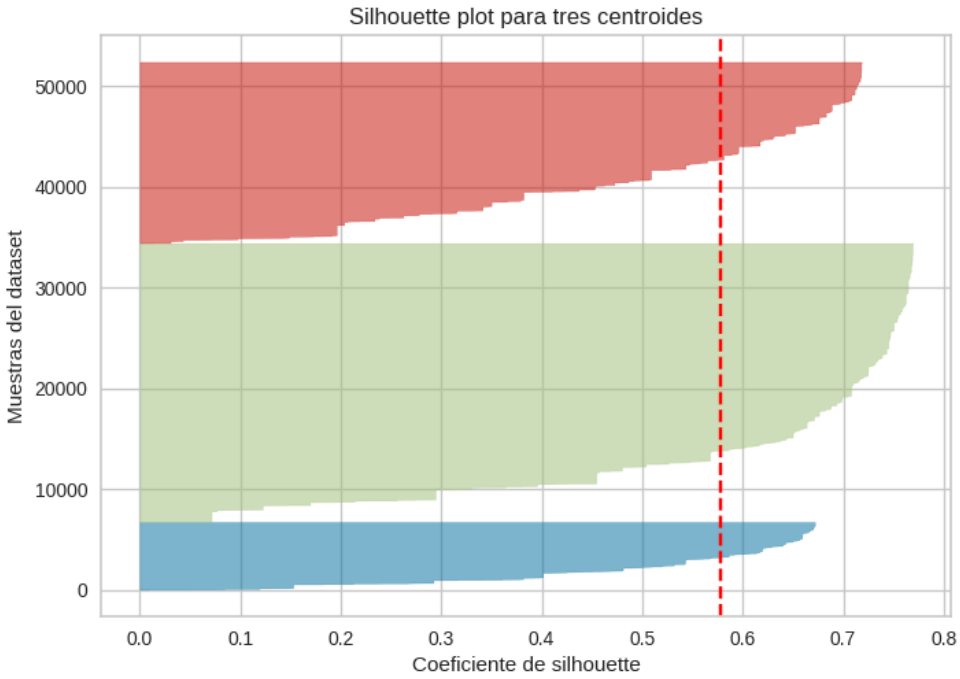
Para calcular la tendencia al agrupamiento de los datos se utilizó la estadística de Hopkins, calculando la misma varias veces y promediando el resultado (recordemos que esta estadística no es determinista). La **Tendencia al agrupamiento resultó de un 99.65%.** Es decir, es casi seguro que el dataset tiene grupos en los datos.

Luego, por medio del entrenamiento de un modelo de k-means probando con distintas cantidades de grupos se llegó a la conclusión de que la cantidad ideal (es decir, el “codo” de la gráfica de error) **es de tres o cuatro grupos** (se utiliza cuatro ya que la consigna pide analizar tres).

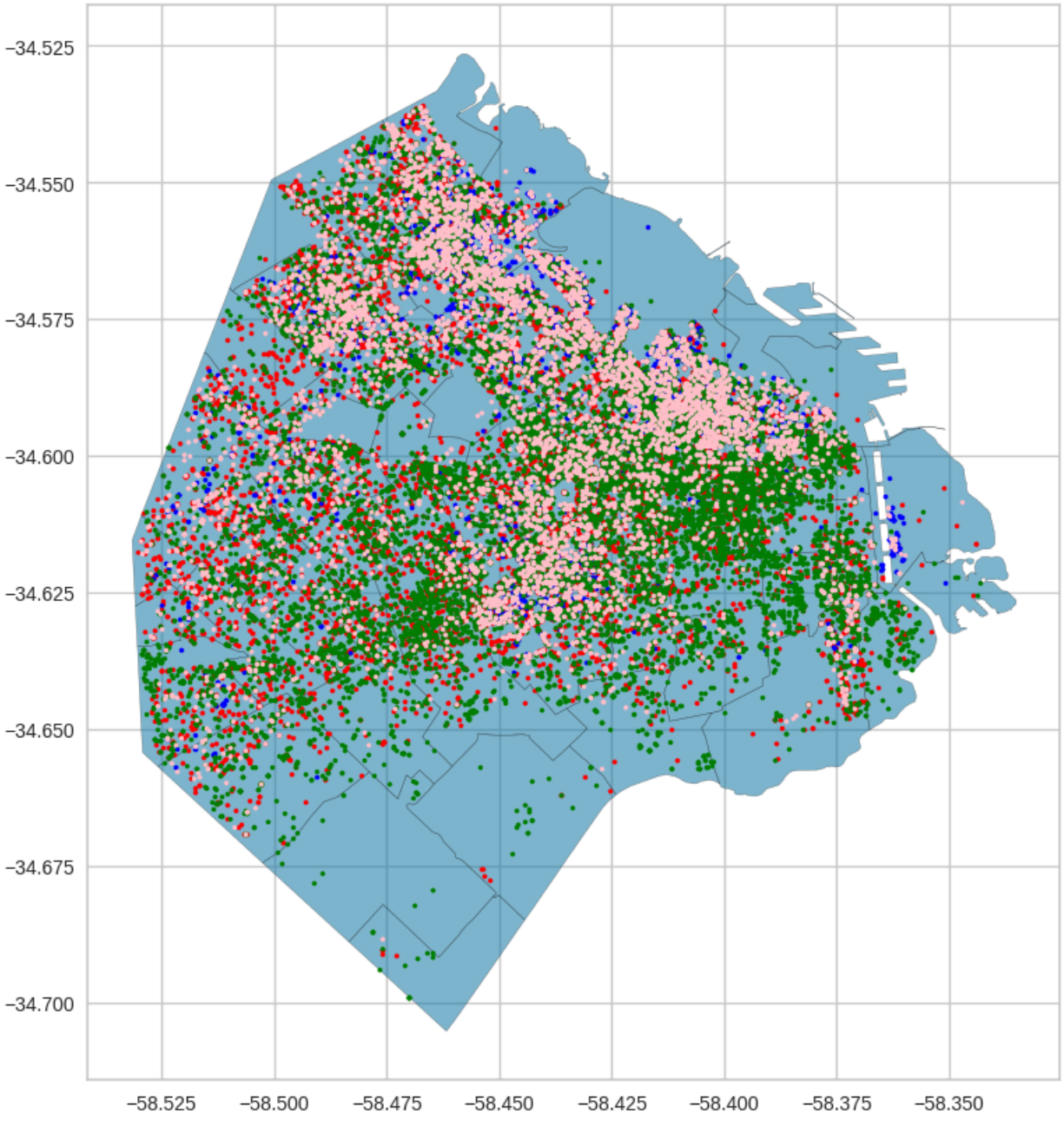
Por grupo, los puntajes de silhouette son .633 para dos clusters, .578 para tres y .569 para cuatro.

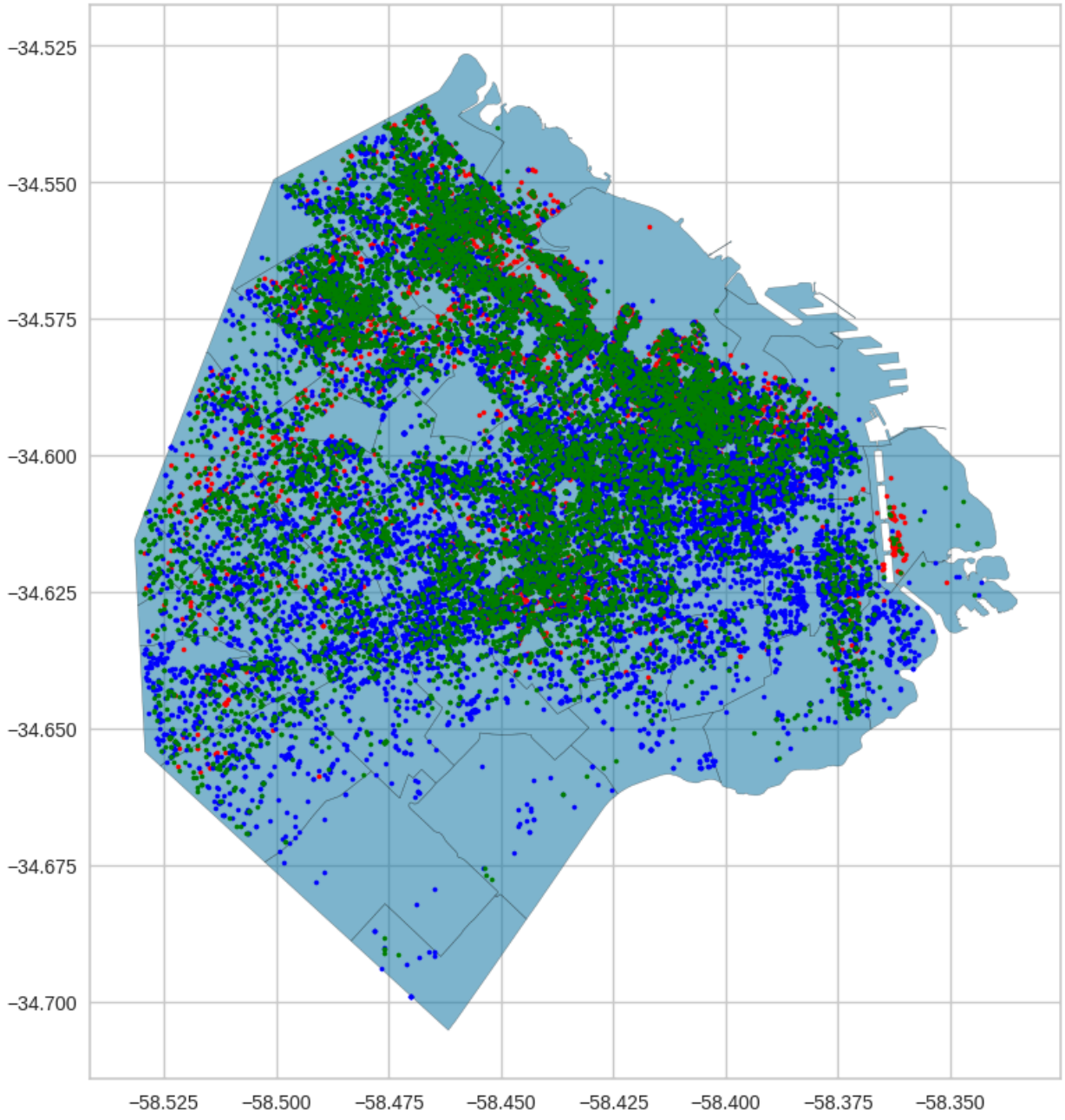

Agrupamiento: Error en función de clusters
1e14
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Error
2
4
6
8
10
12
14
Cantidad de clusters


Silhouette plot para cuatro centroides
50000
40000
30000
20000
10000
0
Grupos
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Coeficiente de silhouette



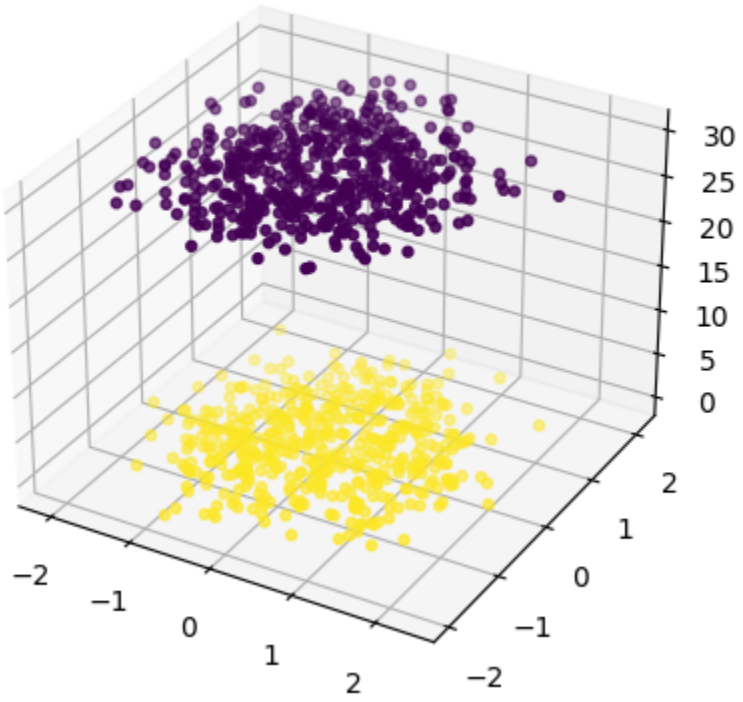
De los gráficos de agrupamiento por el coeficiente de silhouette podemos observar como parecería haber por lo menos dos grandes grupos, que según sean más o menos clusters van variando en tamaño pero no en importancia.
Finalmente, estos grupos en los mapas de CABA resultan de la siguiente manera:


−34.525
−34.550
−34.575
−34.600
−34.625
−34.650
−34.675
−34.700
−58.525
−58.500
−58.475
−58.450
−58.425
−58.400
−58.375
−58.350

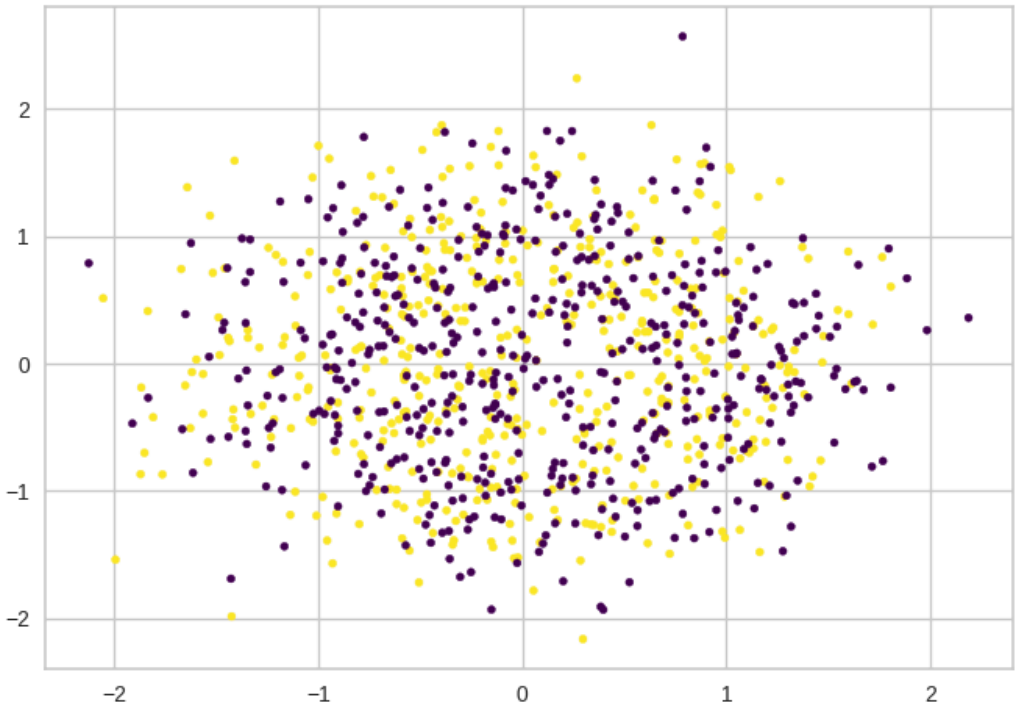


Parecería haber una tendencia al agrupamiento de las propiedades verdes sobre la zona contigua a la costa, mientras que los otros dos grupos no parecen agruparse de forma muy clara.

Al respecto de la falta de claridad del agrupamiento, queremos hacer un pequeño ejemplo para mostrar que es lo que está pasando. Se propone crear un dataset tridimensional conformado por dos discos que se encuentran a diferentes alturas (Hopkins 98.92%):



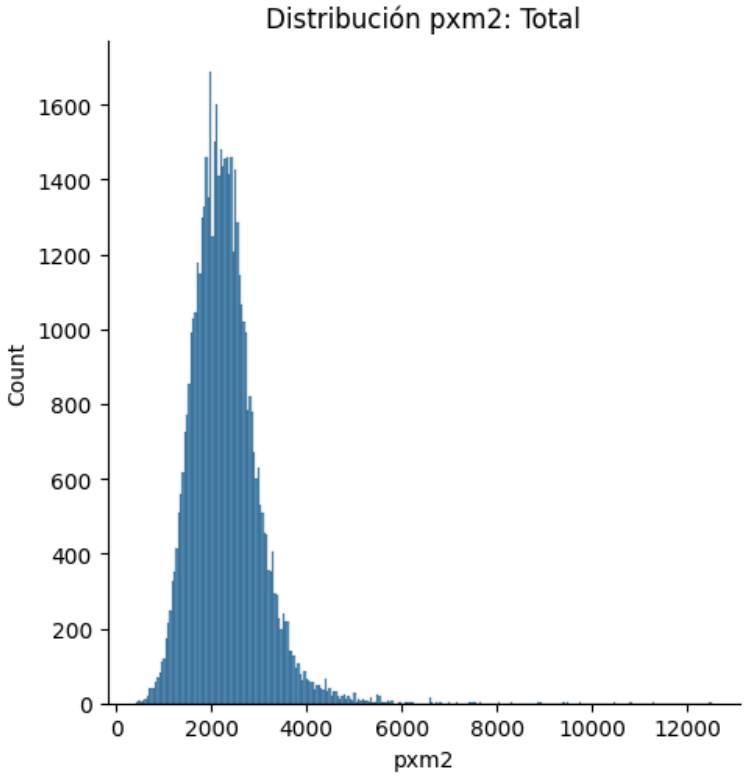
Podemos observar como se encuentran perfectamente agrupados en dos clusters. Pero si yo proyecto el resultado en el plano XY obtengo lo siguiente:

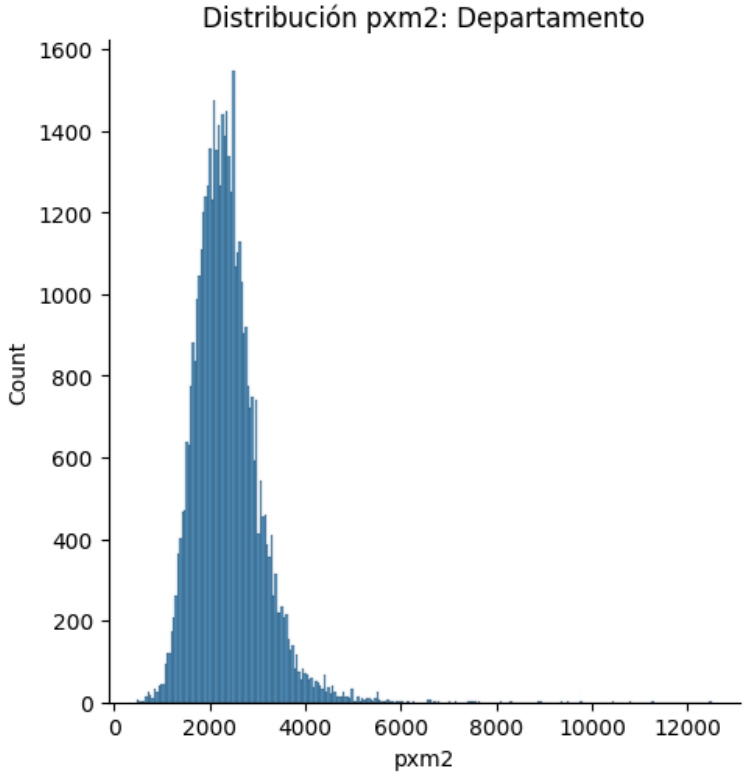


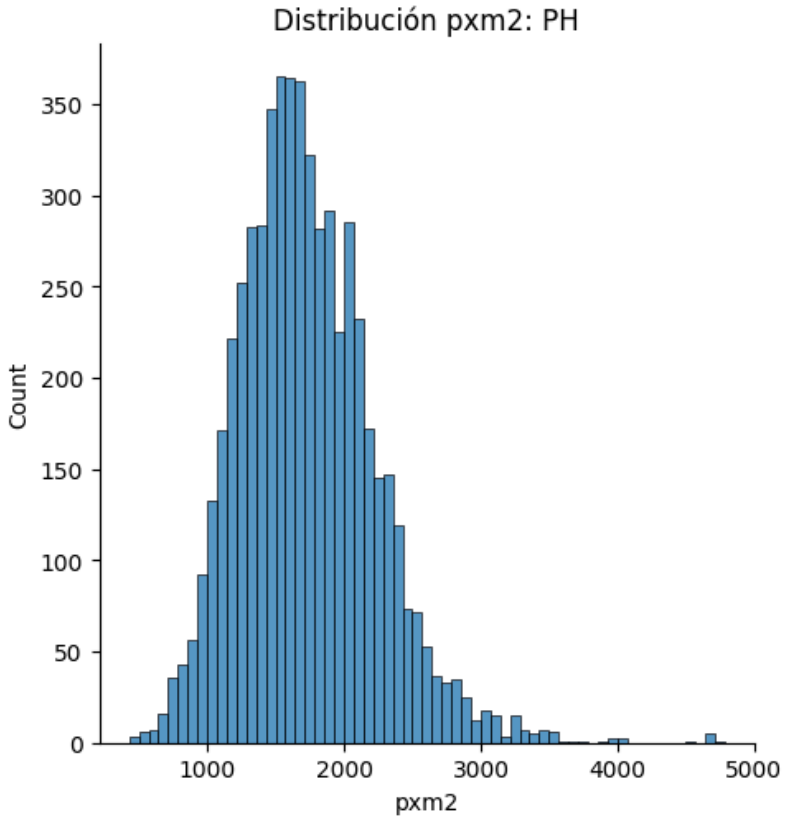
Desde esa proyección parecería que el agrupamiento fue malo, pero nada estaría más lejos de la realidad. El agrupamiento fue muy bueno, pero la proyección es poco representativa del agrupamiento ya que anula la visualización de la dimensión sobre la cual se separan.
Lo que ocurre en nuestros mapas es análogo, donde hay zonas con una clusterización poco clara debido a que estamos proyectando el dataset en dos de las dimensiones menos significativas para el agrupamiento.
En conclusión, latitud y longitud no son las dimensiones que mejor agrupan los datos (por lo menos para esta cantidad de clusters), lo cual ya era sabido debido a la baja correlación que tienen con las demás variables, y que los demás features (precio, superficie, cantidad de habitaciones) resultan de mayor importancia en este punto (seguiremos con este análisis en la sección de clasificación, e inclusive daremos aún más sentido al resultado)

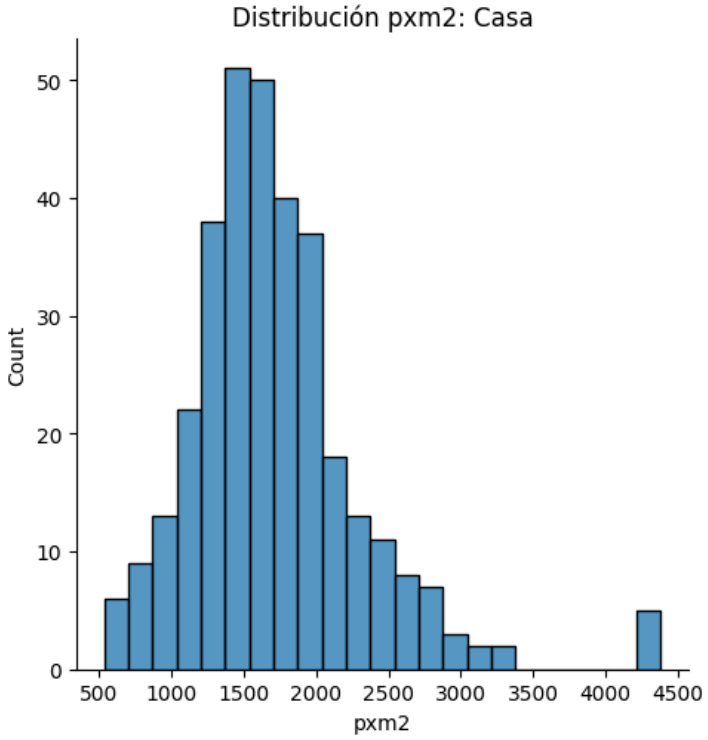
Clasificación

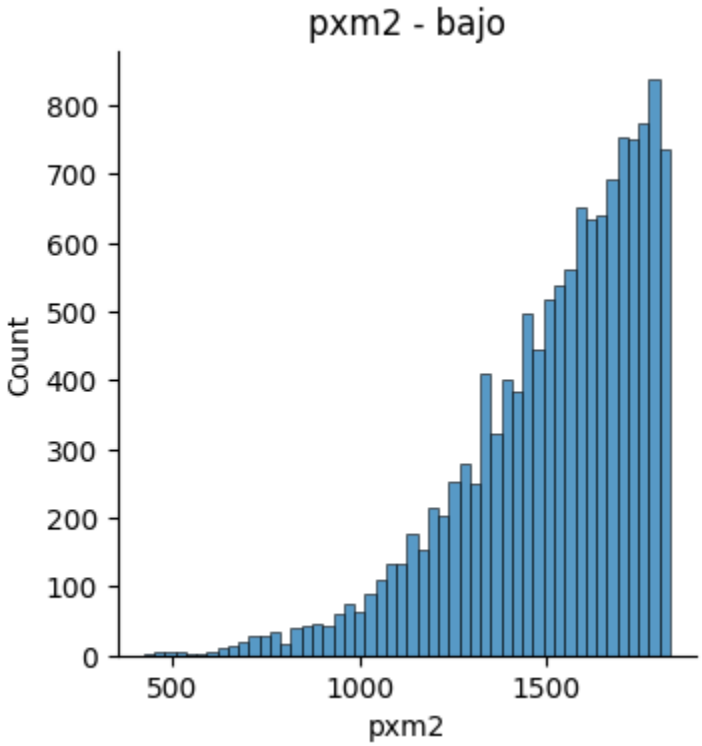
Para la creación de la variable **“tipo_precio” la alternativa elegida es percentiles.** El principal motivo de la elección es la normalidad de los datos, ya que nuestro dataset está compuesto en una gran mayoría por departamentos y, por el poco peso de los demás tipos de propiedades, el resultado es aproximable a una gaussiana. Por lo tanto, lo más recomendable para separar normal de anormal es por medio de desviaciones estándar. En nuestro caso, las divisiones serían de 0 a 16%, de 16% a 84% y de 84% a 100% (derivado de la regla de 68-95-99.7).
Respecto de las otras dos alternativas, ambas fueron descartadas ya que por un lado la separación en tipo de propiedad creemos que no aportaría mucha información, y por el otro la elección de intervalos regulares no es lo más eficiente ya que definimos normalidad en los datos (elegir intervalos regulares sería más certero para una distribución de la cual no tenemos información alguna).
En los siguientes gráficos podremos observar las distintas distribuciones solicitadas. Como podemos observar, se corrobora el gran peso que tienen en el dataset los departamentos y la diferencia de medias y cantidad de muestras.
Luego, en los gráficos de las distribuciones se puede observar la homogeneidad del resultado, donde el precio medio tiene una distribución aproximadamente uniforme, mientras que las dos distribuciones de precio bajo y alto son monótonas decrecientes (coherente con la definición de precio bajo y alto).

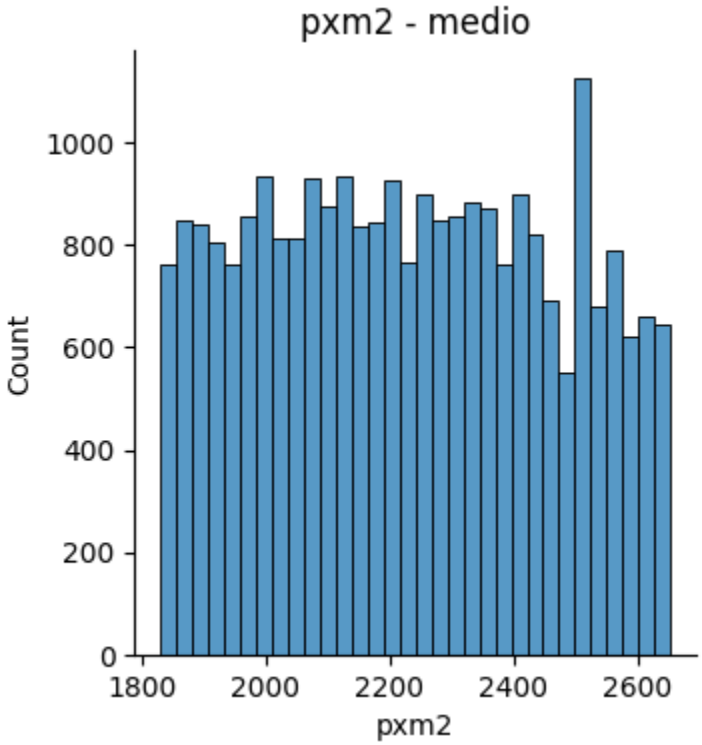

Distribución pxm2: Total
Count
pxm2
0
200
400
600
800
1000
1200
1400
1600
0
2000
4000
6000
8000
10000
12000

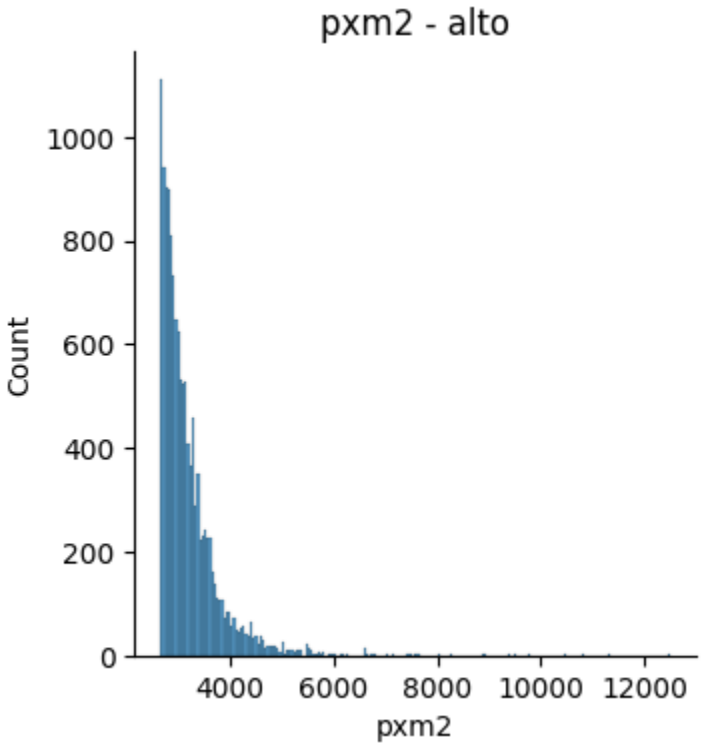

Distribución pxm2: Departamento
Count
pxm2
0
200
400
600
800
1000
1200
1400
1600
0
2000
4000
6000
8000
10000
12000


Distribución pxm2: PH
Count
0
50
100
150
200
250
300
350
1000
2000
3000
4000
5000
pxm2

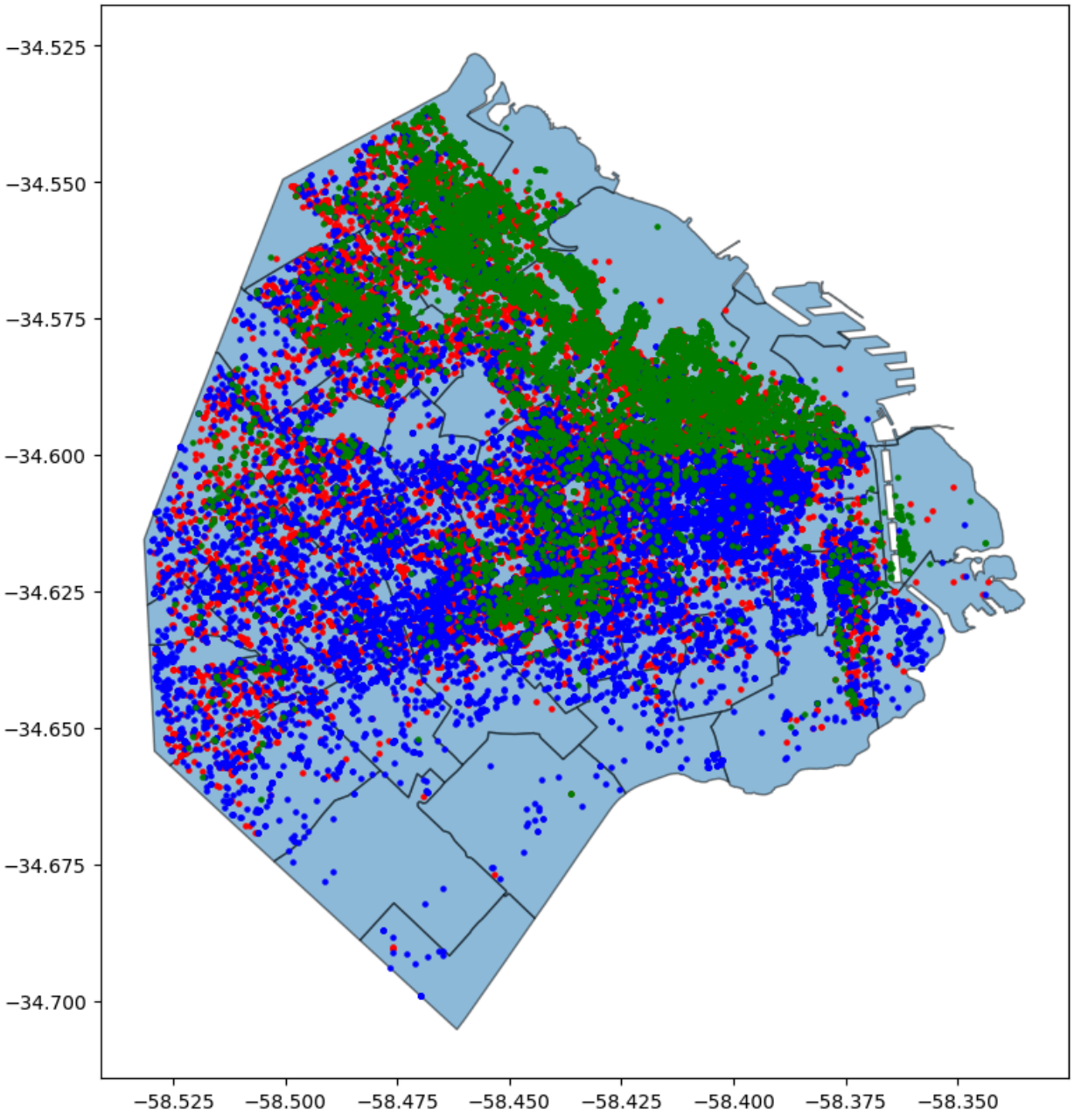

Distribución pxm2: Casa
Count
0
10
20
30
40
50
500
1000
1500
2000
2500
3000
3500
4000
4500
pxm2


pxm2 - bajo
Count
0
100
200
300
400
500
600
700
800
500
1000
1500
pxm2


pxm2 - medio
Count
0
200
400
600
800
1000
1800
2000
2200
2400
2600
pxm2



En el siguiente gráfico podremos observar el gráfico de CABA con las propiedades pintadas en **verde para precio alto**, **rojo para precio medio** y **azul para precio bajo** (colores elegidos para que coincida con mapa de KMeans).
Existe un gran parecido, por lo menos en el orden cualitativo con el resultado del agrupamiento por KMeans, donde la mayor cantidad de propiedades del grupo verde tienen una densidad elevada en la zona contigua al Río de la Plata y en el centro de capital, mientras que las azules y las rojas cubren de forma intercalada el mapa, siendo las azules las que mayor superficie de Capital abarcan.
Vale aclarar que hay una diferencia de cantidad de elementos en los grupos, ya que en KMeans habían quedado un grupo con aproximadamente ocho mil elementos, otro de treinta mil y un último de doce mil, lo cual explica diferencias en menores en los gráficos, considerando que en ciertos puntos el agrupamiento por KMeans podría diferir por la diferencia de dimensionalidad del análisis con respecto a la división por precio por metro cuadrado.

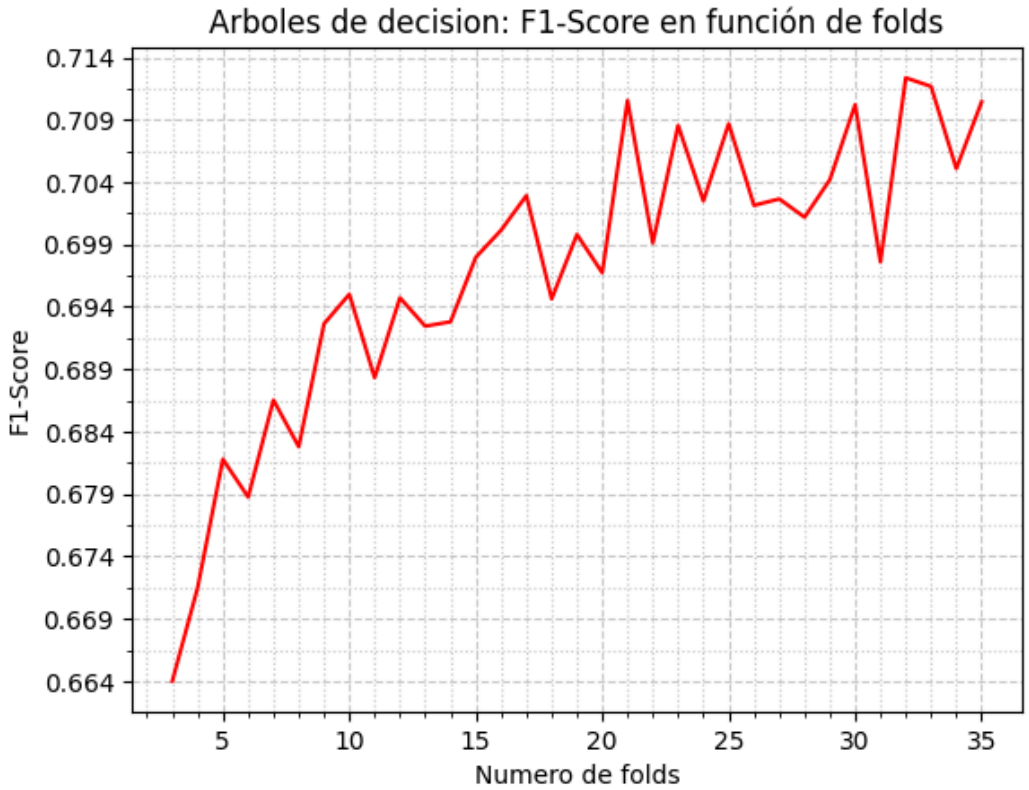

−34.525
−34.550
−34.575
−34.600
−34.625
−34.650
−34.675
−34.700
−58.525
−58.500
−58.475
−58.450
−58.425
−58.400
−58.375
−58.350

Arbol de decisión

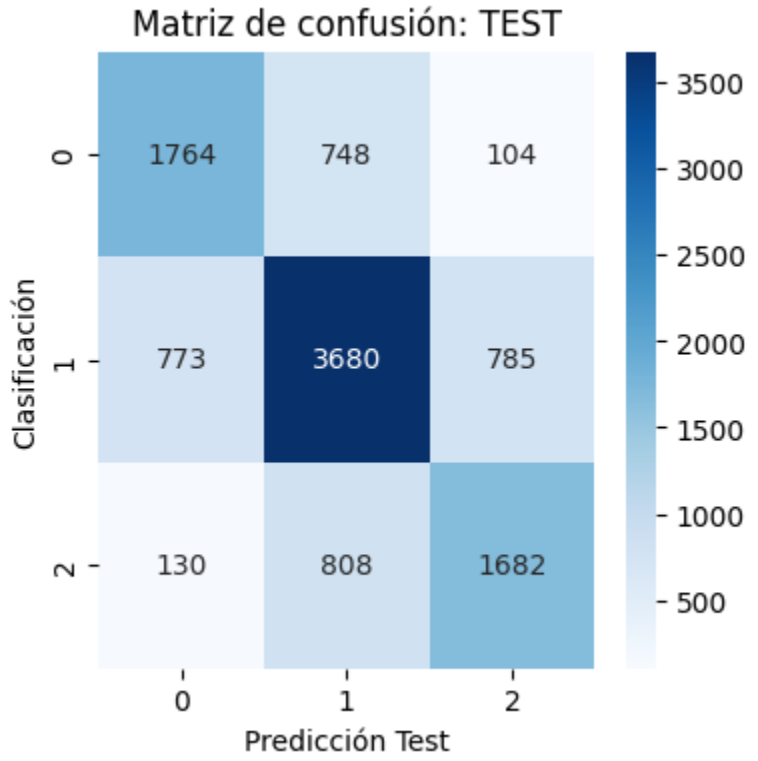
Para el entrenamiento del árbol decisión se decidió utilizar como **métrica el F1-Score**, ya que para nuestro modelo debemos tener en cuenta tanto la precisión de sus clasificaciones (cuanto acierta respecto del total categorizado como una clase) como la exhaustividad en la detección de los positivos en el dataset. Fue elegido beta igual uno debido a que en nuestra consideración el peso de la precisión es igual al de la exhaustividad (en f–beta-score beta representa el peso de la exhaustividad respecto de la precisión).

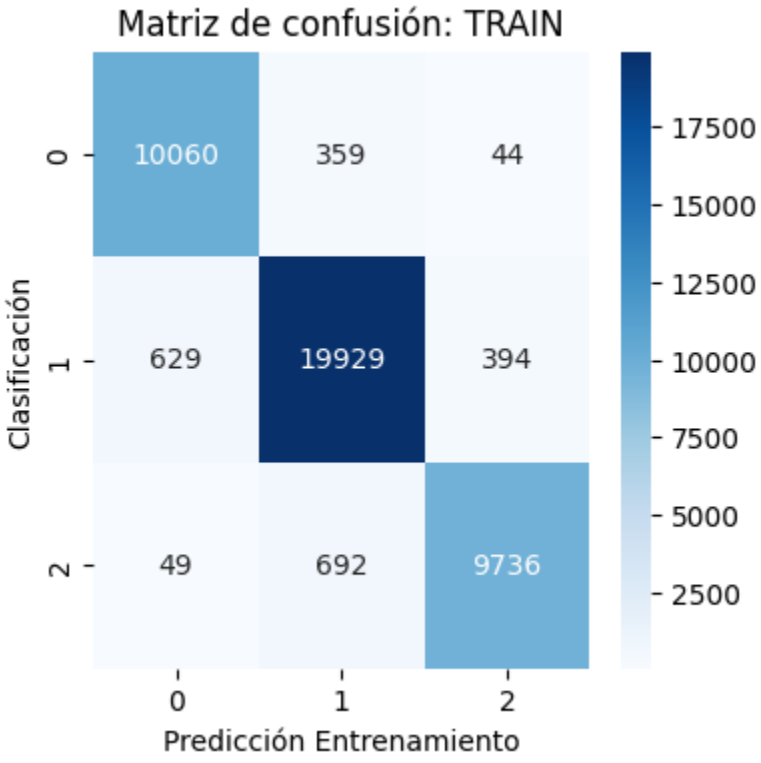
La métrica accuracy fue descartada para el entrenamiento, ya que la misma mide el desempeño global del modelo (total de aciertos respecto del total del dataset) y no es lo más ideal para nuestro caso. Preferimos que el modelo se desempeñe globalmente un poco peor y que categorize algunos negativos como positivo, a un modelo exacto en sus clasificaciones pero que deje pasar algunos positivos con tal de no equivocarse.

La cantidad elegida de folds fue de 10 folds. El motivo es que buscamos un valor que se encuentre a mediados de la tabla, tal que nos encontremos en un punto de equilibrio entre el overfit de los datos de entrada, con una mala generalización, y un underfit con mal desempeño general. Un factor extra en la decisión es que la cantidad de 10 folds es un estándar de la industria.

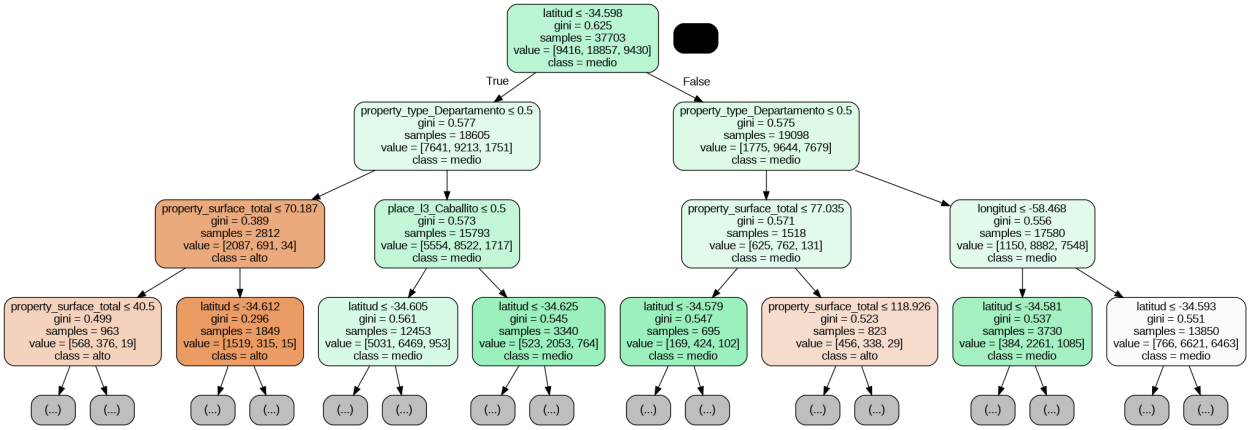


Las matrices de confusión resultantes son las siguientes:





El análisis es que en entrenamiento tenemos un grado de overfit (resulta en un F1-score superior al 90%), pero de todos modos la generalización alcanzada no es para nada mala, oscilando el 70% de precisión y exhaustividad para las clasificaciones de los distintos precios.
Pequeña aclaración, la cantidad de aciertos no es la misma para todas las clasificaciones, pero esto se debe a que el dataset se encuentra dividido en 25-50-25. Como confirmación podemos observar que las muestras bien categorizadas como precio bajo y alto son aproximadamente la mitad de las de precio medio.
Y un punto que queremos notar es que el modelo tuvo muchos menos errores entre las categorías precio bajo-alto que entre categorías contiguas (bajo-medio y medio-alto). Esto es un punto positivo, ya que el modelo resulta mucho menos propenso a cometer un error burdo que a cometer errores más finos.
Las primeras ramas del árbol son las siguientes:



Para explicar las decisiones tomamos la siguiente nomenclatura:
La notación a.b indica "a" decisión (altura descendida en el árbol) y "b" posición de izquierda a derecha. Las pares y el cero corresponden a respuestas verdaderas y las impares a falsas

0. Latitud menor que -34.6 (todo lo que este al sur del obelisco). Divide al dataset en la mitad. Viendo el mapa de CABA es coherente, ya que al norte de esa latitud preponderan las propiedades de precio alto.

1. No es un departamento

2.0 La superficie de la propiedad es menor que 70.2 metros cuadrados.

2.1 La propiedad no esta en Caballito

2.2 La superficie de la propiedad es menor que 77.0 metros cuadrados.

2.3 Longitud menor que que -58.5 (al oeste de la cancha de Velez). Consideramos llamativa la decisión, la que no es de las zonas mas densas del mapa.

De todas formas, vale mencionar que es una zona con mayoría de propiedades de precio bajo y medio.

3.0 La superficie es menor a 40.5 metros cuadrados

3.1 Latitud menor que -34.61 (Al sur de Avenida Belgrano aproximadamente). No comprendemos del todo el motivo, creemos que la banda formada entre esta decisión y la decisión cero es estrecha como para representar un carácter significativo, pero parecería ser un ajuste fino por parte del modelo (esta dividiendo respecto de 1841 muestras).

3.2-3.3-3.4-3.6-3.8 Mismo caso que anterior

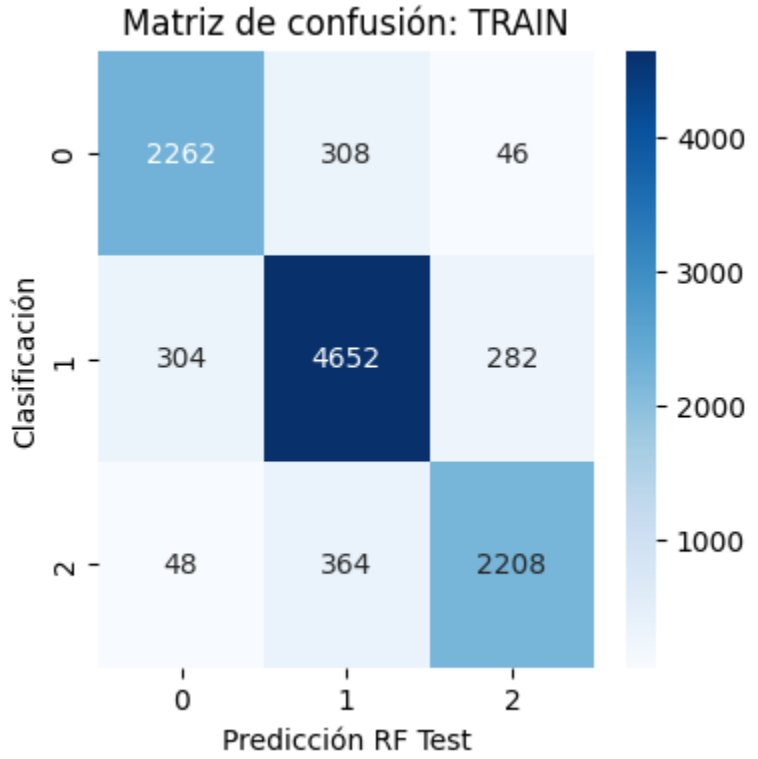
3.5 Superficie menor que 119 metros cuadrados. Es interesante, porque esto permite clasificar en propiedades de precio alto, mientras que los nodos anteriores clasifican en precio medio.

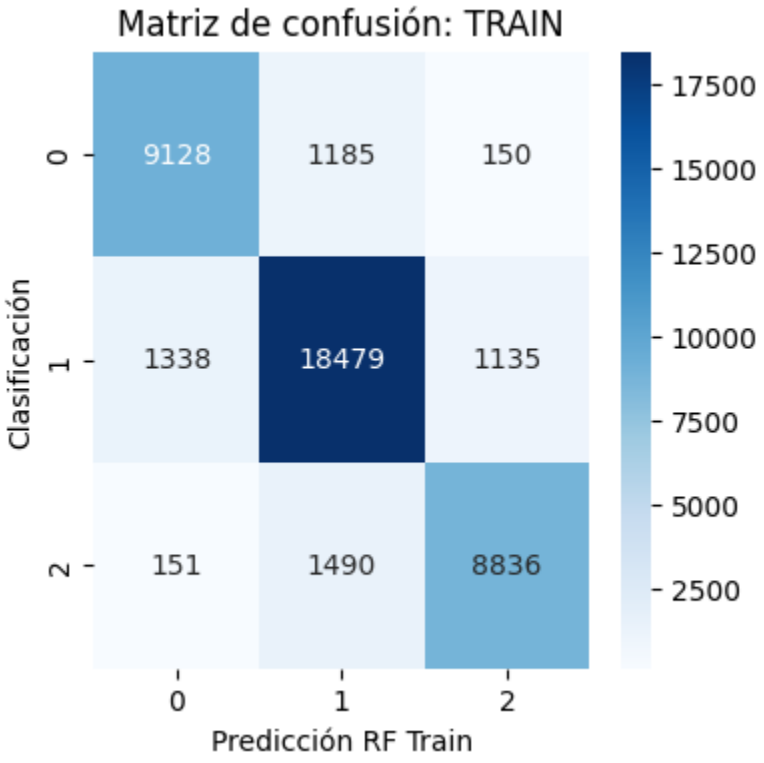
Random forest

Para el entrenamiento de random forest **se utilizaron dos métricas**. Por un lado **F1-Score** (mismos motivos que el caso anterior) y por otro lado se utilizó **ROC AUC**. Esta última debido a que con este modelo queremos que el modelo total pueda balancear correctamente el tradeoff entre Precisión y Recall.

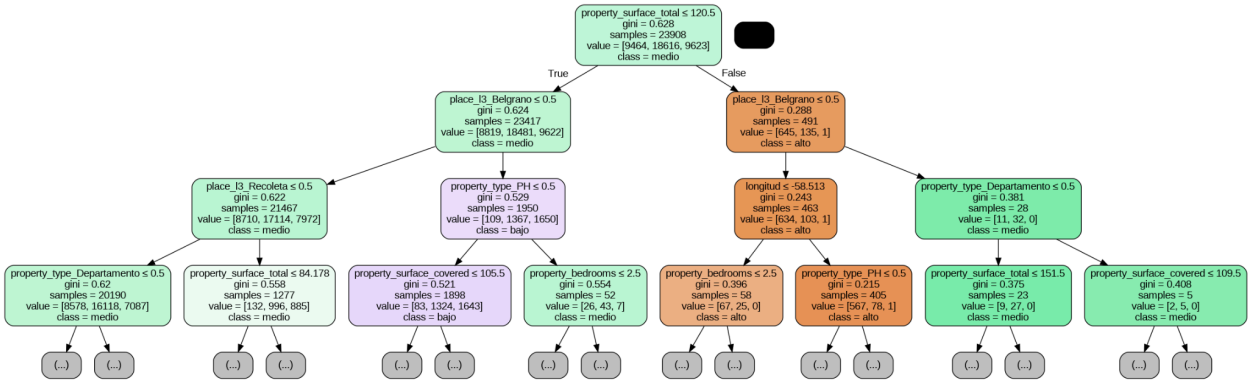
La cantidad de **Folds elegida fue la misma que para el árbol de decisión**, ya que con esto podríamos además comprobar la hipótesis de promediación de estimadores pobres resulta en una mejora general de las predicciones del modelo (una posible técnica contra el overfitting).

Las matrices de confusión resultantes fueron las siguientes:


Matriz de confusión: TRAIN
2262
308
46
304
4652
282
48
364
2208
0
1
2
Clasificación
Predicción RF Test
4000
3000
2000
1000


Matriz de confusión: TRAIN
9128
1185
150
1338
18479
1135
151
1490
8836
0
1
2
Clasificación
Predicción RF Train
17500
15000
12500
10000
7500
5000
2500

La principal conclusión a la que podemos llegar es que el modelo a mejorado significativamente sus predicciones en test, debido a que por Random Forest es un ensamble de tipo bagging donde se promedian los estimadores, lo cual confirma la hipótesis inicial y mejoran las estimaciones del árbol de decisión anterior.
Para el análisis de uno de los árboles que conforman el bosque se eligió el árbol número trece (por cuestiones esotéricas respecto al número en sí, no por algún motivo técnico en particular). El cual pasamos a detallar las decisiones tomadas



0. Superficie de la propiedad menor que 121 metros cuadrados (llamativo porque divide el dataset en 2 y 98% de las muestras).

1. La propiedad no está en Belgrano (divide al dataset en aproximadamente 90 y 10 %)

2.0 La propiedad no está en Recoleta (divide en 95% y 5%)

2.1 La propiedad no es un PH

2.2 La propiedad tiene una longitud menor que -58.5 (está al oeste de la facultad de Agronomía)

2.3 La propiedad no es un departamento

3.0 La propiedad no es un departamento

3.1 La superficie es menor que 84 metros cuadrados

3.2 La superficie cubierta es mayor que 106 metros cuadrados

3.3-3.4 La propiedad tiene menos de tres habitaciones

3.5 La propiedad no es un PH

3.6 La superficie es menor que 152 metros cuadrados 3.7 La superficie cubierta es menor que 110 metros cuadrados

Lo más llamativo de este árbol es que no parecería estar dividiendo de forma homogénea al dataset. En sí mismo no tiene por qué ser un problema, ya que el proceso de creación de un árbol consta en la toma de ciertas decisiones basadas en sus atributos tal que las divisiones formen grupos con la mayor cantidad de muestras de una misma clase, lo cual puede llevar tanto a la conformación de

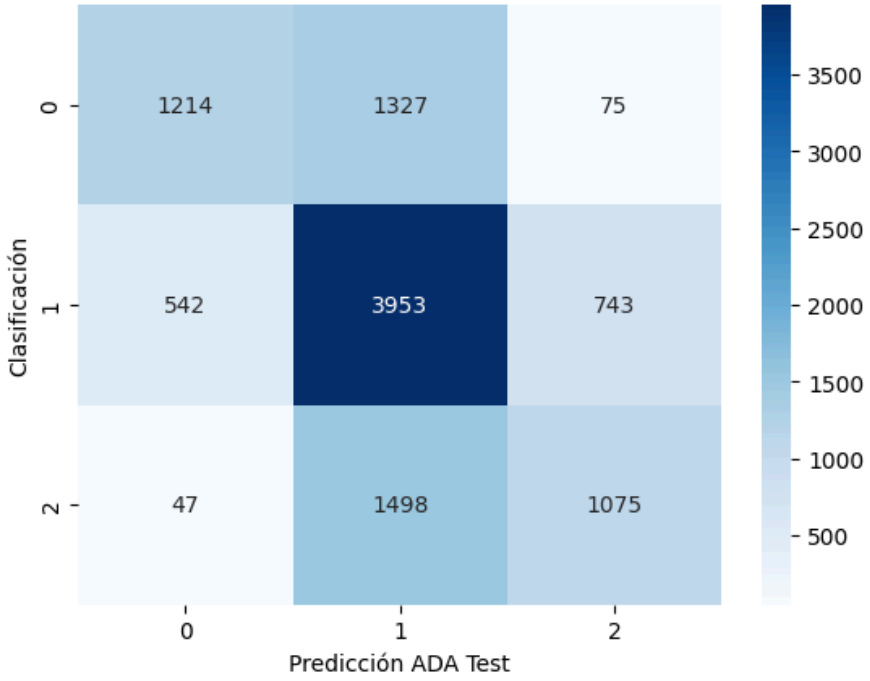
grandes grupos con distintas clases, o pequeños grupos con relativa homogeneidad (decisión tomada en función de Gini o de la entropía.

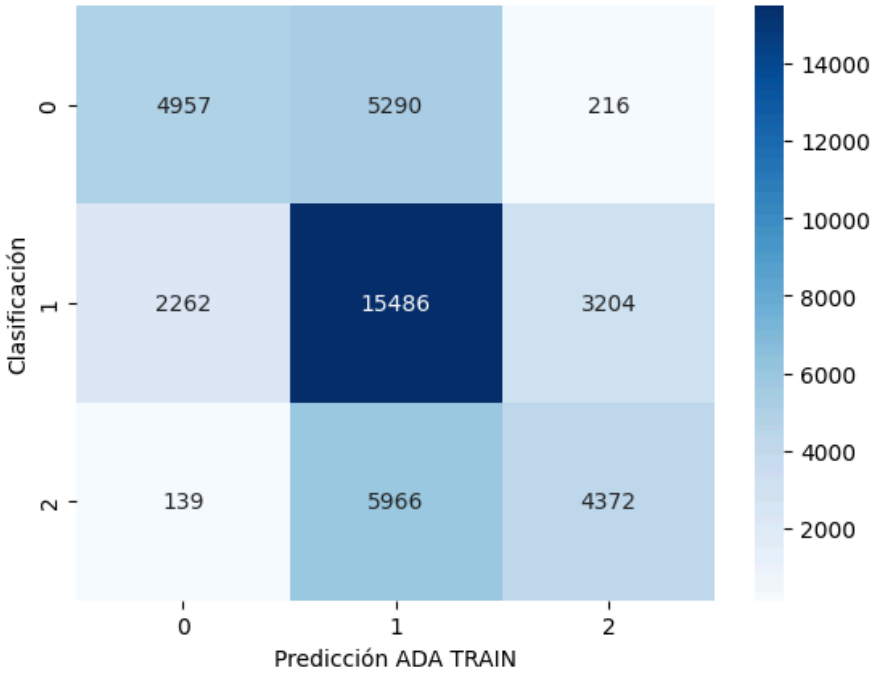
AdaBoost

Tratándose de un problema de clasificación en base a atributos numéricos y no numéricos, decidimos que mantener la línea de los árboles de decisión era lo más correcto. (sumando además una coherencia con el trabajo que se viene realizando en esta sección, permitiendo la comparativa entre modelos resultantes de un fundamento similar).

Los resultados del entrenamiento resultaron sorpresivos, ya que el modelo tuvo una performance inclusive peor que el árbol de decisión. Lo correcto es que no comete errores burdos, pero es más propenso que los modelos anteriores a equivocarse en las clasificaciones contiguas (precio bajo-medio, medio-alto).

El principal motivo de este desempeño creemos que se trata del bajo recall, el cual empeora la performance global y





Métricas generales

Modelo	F1-Test	Precision Test	Recall Test	Accuracy Test	Roc-Auc Score Test
Arbol de Decision	0.68	0.67	0.68	0.68	-
Random Forest	0.895	0.895	0.895	0.89	0.87
AdaBoost	0.60	0.60	0.67	0.59	-

Regresión

Si llegaron a entrenar alguno de los modelos, mencionar cuáles y qué métricas obtuvieron en test y si realizaron nuevas transformaciones sobre los datos (encoding, normalización, etc) completando los ítems a y b:

a. Construcción del modelo

KNN

- ¿Utilizaron K-fold Cross Validation? ¿Cuántos folds utilizaron? Utilizamos K-fold CV, utilizando 5 folds.
- ¿Qué métrica utilizaron para buscar los hiper parámetros? Utilizamos R2 para la búsqueda de hiperparametros.
- Evaluar la performance del modelo en el conjunto de evaluación, explicar todas las métricas. Comparar con la performance de entrenamiento.

Utilizamos Random Search Cross Validation con 15 iteraciones para buscar los mejores hiperparametros. Seleccionando entre las siguientes variables:

- 'leaf_size': list(range(1, 40))
- 'n_neighbors': list(range(2, 40))
- 'p': [1, 2, 3, 4]
- 'weights': ['uniform', 'distance']

De donde obtuvimos los siguientes resultados:

- 'weights': 'distance'
- 'p': 1
- 'n_neighbors': 21
- 'leaf_size': 34

En cuanto al funcionamiento del modelo, obtuvimos las siguientes métricas:

Resultados contra los datos de test:
> R2 = 0.8395
> RMSE = 25308.5719
> RMSE = 640523810.8002

Resultados contra los datos de train:
> R2 = 0.9947

> RMSE = 4506.4962
> RMSE = 20308507.9679

Resulta evidente que el modelo está realizando un severo overfitting, lo cual salta a la vista tanto por el elevado valor de sus métricas al trabajar con los datos de train, como así también por la considerable variación de dichas métricas al enfrentarse al conjunto de test.

XGBoost

- ¿Utilizaron K-fold Cross Validation?¿Cuántos folds utilizaron? Utilizamos K-fold CV, utilizando 5 folds.
- ¿Qué métrica utilizaron para buscar los hiperparámetros? Utilizamos R2 para la búsqueda de hiperparametros.
- Evaluar la performance del modelo en el conjunto de evaluación, explicar todas las métricas. Comparar con la performance de entrenamiento.

Utilizamos Random Search Cross Validation con 15 iteraciones para buscar los mejores hiperparametros. Seleccionando entre las siguientes variables:

- "learning_rate": np.linspace(0.05, 0.5, 50)
- "gamma": [0, 1, 2]
- "max_depth": list(range(2, 10))
- "subsample": np.linspace(0, 1, 20)
- "lambda": [0, 1, 2]
- "alpha" : [0, 1, 2]
- "n_estimators": list(range(10, 161, 10))

De donde obtuvimos los siguientes resultados:

- 'subsample': 0.6842105263157894
- 'n_estimators': 90
- 'max_depth': 7
- 'learning_rate': 0.20612244897959187
- 'lambda': 1
- 'gamma': 1
- 'alpha': 2

En cuanto al funcionamiento del modelo, obtuvimos las siguientes métricas:

Resultados contra los datos de test:

> R2 = 0.809

> RMSE = 27610.2765

> RMSE = 762327366.7294

Resultados contra los datos de train:

> R2 = 0.866

> RMSE = 22726.6807

> RMSE = 516502014.6365

Podemos ver que en este modelo tenemos una clara mejora con respecto a KNN, ya que no solo tenemos una métrica considerablemente más alta en el manejo de los datos de test, sino que también tenemos resultados mucho más equilibrados para las métricas al comparar los resultados en test y train.

Modelo a elección (Support Vector Machines)

- ¿Utilizaron K-fold Cross Validation?¿Cuántos folds utilizaron? Utilizamos K-fold CV, utilizando 5 folds.
- ¿Qué métrica utilizaron para buscar los hiperparámetros? Utilizamos R2 para la búsqueda de hiperparametros.
- Evaluar la performance del modelo en el conjunto de evaluación, explicar todas las métricas. Comparar con la performance de entrenamiento.

Utilizamos Random Search Cross Validation con 15 iteraciones para buscar los mejores hiperparametros. Seleccionando entre las siguientes variables:

- "C": [100, 10, 1, 0.75, 0.5]
- "loss": ["epsilon_insensitive", "squared_epsilon_insensitive"]
- "max_iter": list(range(1000, 10000, 2000))

De donde obtuvimos los siguientes resultados:

'max_iter': 7000
'loss': 'squared_epsilon_insensitive'
'C': 10

En cuanto al funcionamiento del modelo, obtuvimos las siguientes métricas:

Resultados contra los datos de test:
> R2 = 0.6233
> RMSE = 38775.6268
> RMSE = 1503549237.2359

Resultados contra los datos de train:
> R2 = 0.636
> RMSE = 37451.6264
> RMSE = 1402624320.5654

En este caso, tenemos resultados un poco más desalentadores. Si bien se mantiene un buen equilibrio entre las métricas de train y test, tenemos resultados inferiores incluso a los obtenidos en KNN al trabajar con los datos de test.

b. Cuadro de Resultados

Realizar un cuadro de resultados comparando los modelos que entrenaron (entre ellos debe figurar cuál es el que seleccionaron como mejor predictor).

Medidas de rendimiento en el conjunto de TEST:

- MSE
- RMSE
- R2

Confeccionar el siguiente cuadro con esta información:

Modelo	MSE Test	RMSE Test	R2 Test
KNN	640523810	25308	0.8395
XGBoost	762327366	27610	0.809
SVM	1503549237	38775	0.6233

En cada caso ¿Cómo resultó la performance respecto al set de entrenamiento?

KNN: Tuvo una excelente performance al trabajar con el conjunto de entrenamiento (Superior a un R2 = 0.99) lo cual era un claro indicador de que el modelo estaba haciendo un overfitting.

XGBoost: Tuvo un desempeño bastante similar al obtenido luego con el conjunto de test (Rondando un R2 = 0.8)

SVM: Tuvo un desempeño similar al obtenido luego con el conjunto de test, pero bastante pobre (Rondando un R2 = 0.6)

Nota: indicar brevemente en qué consiste cada modelo de la tabla

KNN: Este método de regresión encuentra la predicción para un conjunto de datos basándose en la media de los valores de los k vecinos más cercanos.

XGBoost: Este método de regresión utiliza un conjunto de árboles de decisión débiles para hacer predicciones. Utiliza la técnica de refuerzo (boosting), donde los árboles se construyen secuencialmente, y cada árbol intenta corregir los errores cometidos por los árboles anteriores.

SVM: En la regresión SVM, el objetivo es encontrar la función que mejor separe los puntos de datos mientras minimiza la violación del margen.

Estado de Avance

1. Análisis Exploratorio y Preprocesamiento de Datos

Porcentaje de Avance: 100%/100%

Tareas en curso: trabajo terminado.

Tareas planificadas: ninguna.

Impedimentos: ninguno.

a) Exploración Inicial: análisis concluido.
b) Visualización de los datos: análisis concluido.

c) Datos Faltantes: análisis concluido.

d) Valores atípicos: análisis concluido.

e) Opcional: -

2. Agrupamiento

Porcentaje de Avance: 100%/100%

Tareas en curso: trabajo terminado.

Tareas planificadas: ninguna.

Impedimentos: No logramos corregir el error de Slack para la estadística de Hopkins, utilizamos una función para el cálculo manual (detalles en notebook)

3. Clasificación

Porcentaje de Avance: 100%/100%

Tareas en curso: trabajo terminado.

Tareas planificadas: ninguna.

Impedimentos: ninguno.

4. Regresión

Porcentaje de Avance: 100%/100%

Tareas en curso: trabajo terminado.

Tareas planificadas: ninguna.

Impedimentos: ninguno.

Tiempo dedicado

Indicar brevemente en qué tarea trabajó cada integrante del equipo durante estas semanas. Si trabajaron en las mismas tareas lo detallan en cada caso (como en el ejemplo el armado de reporte). Deben indicar el promedio de horas semanales que

dedicaron al trabajo práctico. En esta tabla solo deben incluir las tareas que realizaron luego de entregar el CHP1.

Integrante	Tarea	Prom. Hs Semana
Testa, Santiago Tomas	Clustering y Clasificación	13
Pratto, Federico Nicolás	Regresion	5
Ramirez, Jose Israel	-	-
Torres, Santiago/Danny	-	-