Entrega Final - Grupo 07

Análisis Exploratorio:

El dataset provisto por la empresa Properati consta 460154 registros de propiedades, con 20 columnas detallando ubicación geográfica, precio, ambientes, entre otras características de cada una de ellas. A nuestro parecer los features que más se destacan son “place_l2” (posteriormente renombrada a “Barrio”), su “property_price” (Precio), “property_rooms” (Ambientes) y “property_type” (Tipo de Propiedad).

Una de las características más importantes a la hora de comprar una propiedad es su ubicación. En el data frame trabajado aparece de varias formas, una de ellas mediante coordenadas geográficas (según Latitud y Longitud) y otra es según varias divisiones de donde se encuentra (aunque estén marcadas en varias columnas, Ejemplo: Argentina, Capital Federal, Palermo, Palermo Chico). En todos estos casos el tipo de dato es objeto.

Otra feature a tener muy en cuenta es el precio de la propiedad, donde lógicamente, es un número (float) que al filtrar el dataset según las cuestiones solicitadas, solo trabajaremos los precios en dólares. Sin olvidarnos del tipo de propiedad, que toda persona que hoy en día quiere comprarse una, debe tener preferencias en si quiere un Departamento o una Casa, por ejemplo.

Y por último, pero no menos importante, la cantidad de ambientes que posee la propiedad. Es una característica fundamental a la hora de buscar propiedades a adquirir, ya que, como luego mostraremos, es una variable muy relacionada con el precio. Cuando una aumenta, en líneas generales, la otra también tiende a subir.

Preprocesamiento de Datos:

1) Cuando comenzamos a analizar el dataset y las variables del mismo, notamos que habían columnas que, en un principio, eran prescindibles asique nos metimos más en profundidad.

   Primero, vemos el porcentaje de nulos de las variables. Vimos que el 100% de los datos de las variables “place_l5” y “place_l6” son nulas. En conclusión, no son relevantes para el análisis y tomamos la decisión de eliminarlas.

   Luego, gracias a los filtrados que fueron solicitados, tenemos tres variables que en realidad son constantes, es decir, tienen un solo valor:

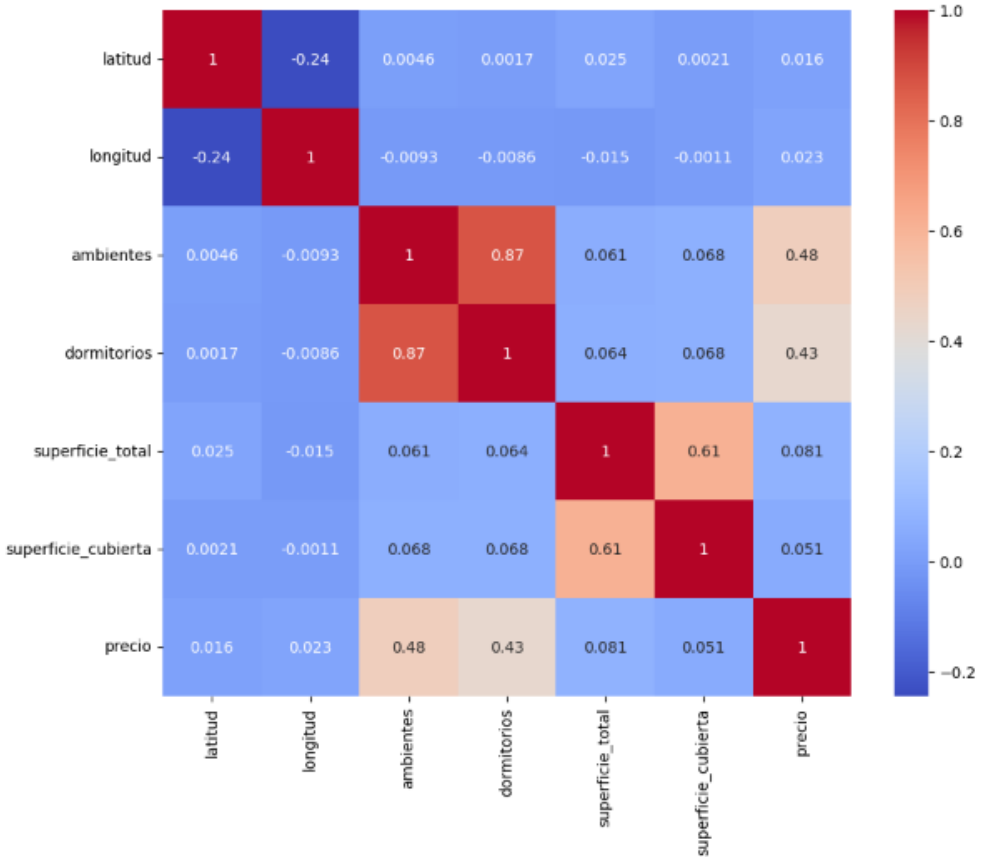
   - “place_l2” sólo contiene el valor “Capital Federal”

- “operation” sólo contiene el valor “Venta”
- “property_currency” sólo contiene el valor “USD” (Dolares)

Por lo tanto, considerando que ya sabemos el valor constante de estas variables en todas las propiedades del dataset, concluimos que no aportan ninguna información adicional para que valga la pena dejarlas en el mismo y procedimos a eliminarlas.

También, en base a los nombres de las columnas, notamos que “start_date” y “created_on” pueden ser similares o hasta iguales. Así que creamos una función que itere en el dataframe y compare todos los valores de ambas las variables en todas las propiedades. En caso de que devuelva True, brindan la misma información, es decir, son iguales, y podemos eliminar una de ellas. Efectivamente, fue así. Por lo tanto la columna “created_on” es redundante y fue eliminada.

2) En un principio, se puede suponer alguna correlación entre variables que suene lógico. Por ejemplo, uno esperaría que cuanto más ambientes ó más metros cuadrados tenga una propiedad, más caro será su precio. Así que procedemos a buscar rápidamente esas posibles correlaciones lineales entre las variables cuantitativas mediante la correlación de Pearson.



Con el gráfico, observamos que la mayoría de las variables no están relacionadas entre sí y aportan mucha información al dataset, sin embargo existen algunas relaciones lineales:

- Ambientes y dormitorios: Una correlación evidente ya que un dormitorio es un ambiente, lo que la hace la correlación lineal más fuerte del gráfico.
- Superficie total y superficie cubierta: Otra correlación evidente ya que la superficie cubierta forma parte de la superficie total, pero a pesar de esto, no es una correlación lineal tan fuerte.
- Precio y ambientes: Una correlación lógica, que por transitividad se traslada también a la correlación entre el precio y los dormitorios. De estas tres relaciones destacadas es la más débil, lo cual no es raro en si, pero si es raro que no tengan una correlación lineal un poco más fuerte.

Por último, llama la atención que distintas variables no tengan una correlación lineal como el precio y la superficie, o por otra parte, la superficie y la cantidad de ambientes.

3) Por el momento, no percibimos la necesidad de crear una nueva columna o feature. Con las ya presentes, se logra analizar el dataset de una forma correcta. En caso de ser necesario, será una opción a tener en cuenta.

5) Aquí, hemos realizado un análisis creando un pequeño dataframe con la cantidad y sus porcentajes de los datos faltantes de cada columna.

	variable	porcentaje	cantidad
id	id	0.000000	0
fecha_inicio	fecha_inicio	0.000000	0
fecha_fin	fecha_fin	0.000000	0
latitud	latitud	3.950981	2979
longitud	longitud	3.950981	2979
barrio	barrio	0.432366	326
subdivision	subdivision	96.193583	72529
tipo	tipo	0.000000	0
ambientes	ambientes	1.119378	844
dormitorios	dormitorios	11.673895	8802
superficie_total	superficie_total	5.159220	3890
superficie_cubierta	superficie_cubierta	3.569013	2691
precio	precio	0.000000	0
nombre	nombre	0.000000	0

Como vemos en la imagen adjunta, hay un par de variables (columnas en el dataset original) que tienen un 0% de nulos en base al total, es decir, es una columna completa en cuanto a datos; como lo es el caso del “id”, de las fechas tanto de inicio como de fin, el precio, el nombre (una breve descripción acerca de la propiedad) y el tipo de propiedad. Luego hay casos donde tienen poco porcentaje de nulos, como lo es la cantidad de “ambientes” de la propiedad (1,1%) o el “barrio” donde se encuentran (0,4%).

Relacionado a este último, vemos una cantidad mayor, aunque siga siendo baja, cantidad de nulos en las coordenadas geográficas, “latitud” y “longitud” (casi un 4%). Antes de ello, observamos que la cantidad de registros que contienen “latitud”, “longitud” y “barrio” nulos, simultáneamente, suman 132. Esta magnitud comparada con el total de registros (75399) de nuestro data frame actual, la consideramos despreciable y esas propiedades serán eliminadas. Luego, para la imputación de los datos faltantes hemos decidido usar un geocodificador mediante un dataset de barrios en CABA propiciado por el gobierno. De esta forma, hemos iterado en todo nuestro dataset a trabajar (el de properati) para ver los “barrios” de cada propiedad e imputarles sus respectivas latitudes y longitudes. Hemos tomado esta convención de “creerle” al barrio donde pertenecen las propiedades por sobre a sus coordenadas por 2 grandes motivos: el 1ero ya que al realizar un gráfico del mapa de CABA, nos percatamos que muchas coordenadas son erróneas e incluso otras están por fuera del rango de CABA. Y el 2do motivo, es que tomando las coordenadas para imputar sus respectivos barrios hubiéramos necesitado un geocodificador inverso y esto hubiera sido un poco más costoso en implementación. Por ende, procedimos a eliminar las propiedades con “barrio” nulo, además de imputar todas las coordenadas a los demás.

Siguiendo con la lista de las variables, observamos que las columnas “superficie total” y “superficie cubierta” tienen porcentajes distintos de nulos en el dataset, marcando casi 5,2% y 3,6% respectivamente. Gracias a que estas columnas tienen una correlación bastante alta, para estos datos faltantes lo que decidimos hacer es realizar un método multivariado de imputación con Regresión Lineal para que se estudien e imputen ambas variables entre sí. De todas formas, antes de seguir realizamos un chequeo de que la “superficie cubierta” no debe ser mayor que la “superficie total” así que en esos casos donde sí sucedía, calculamos el promedio de “superficie cubierta” en todas las propiedades que tengan X cantidad de “superficie total” y fuimos imputando esos promedios de “superficie cubierta” a cada propiedad que tengan esa cantidad X de “superficie total”. Al final de este algoritmo, todas las propiedades deberían cumplir que su “superficie total” es mayor a su “superficie cubierta” o en su defecto, los empatamos para que todos los valores sean coherentes. Posdata: luego de este algoritmo hemos chequeado y eliminado todas las propiedades que tengan una

“superficie total” menor a 20. Ya que, creemos que ninguna propiedad “real” puede tener menos de 20 mts^2.

Y por último, las columnas “ambientes” y “dormitorios” también tienen porcentajes distintos de nulos en el dataset, marcando 1,1% y 11,6% respectivamente. Estas columnas, también tienen una correlación bastante alta, pero en su defecto, aquí no hemos utilizado el método de Regresión Lineal debido a que, generalmente, imputaba valores muy cercanos entre sí o incluso idénticos. Creemos que no todas las propiedades “reales” tienen la misma cantidad de “ambientes” y “dormitorios”. Por lo tanto, realizamos el mismo algoritmo que con las superficies. Calculamos el promedio de “dormitorios” en todas las propiedades que tengan X cantidad de “ambientes” y fuimos imputando esos promedios de “dormitorios” a cada propiedad que tengan esa cantidad X de “ambientes”. Al final de este algoritmo, todas las propiedades deberían cumplir que su “ambientes” es mayor a su “dormitorios” o en su defecto, serán valores iguales pero no tantos como si hubiéramos imputado mediante la Regresión Lineal.

Una vez llegado a este párrafo de texto, habríamos completado todo el dataset con valores coherentes habiendo analizado cada columna en particular, imputando mediante, promedios, medianas, predecidos con la Regresión Lineal o utilizando un geolocalizador.

4) Sí, se han encontrado muchos valores atípicos (Outliers) a lo largo de nuestro análisis del data set. Los pasaremos a explicar en el siguiente orden: Coordenadas, Ambientes y Dormitorios, Superficies (Totales y Cubiertas) y Precio.

Para el caso de las coordenadas, “latitud” y “longitud”, (como hemos mencionado), al realizar un gráfico del mapa de CABA, nos percatamos que muchas coordenadas son erróneas e incluso otras están por fuera del rango de CABA. Las que son erróneas las hemos modificado (en el punto de imputación) y las que están por fuera del rango de CABA las hemos analizado como Outliers. A las propiedades con esta característica, hemos tomado la decisión de buscar a mano en el dataset propiciado por el gobierno, los centroides de cada coordenada de su respectivo barrio. De esta forma, imputamos esos centroides en la “latitud” y “longitud” de la propiedad que “estaba por fuera de CABA” respetando y guiándonos por su “barrio”, de la misma manera que hemos hecho en el punto de imputación.

Para el caso de “ambientes” y “dormitorios”, hemos graficado tanto boxplots para ambas variables como un scatter plot (Dispersión) para observar todas las propiedades de nuestro dataset. Con ellos, a simple vista se veían claros Outliers de casos donde los “ambientes” eran muy altos pero los “dormitorios” eran muy bajos. Lo “ideal” sería que el gráfico de

Dispersión se vea similar a una recta diagonal (matemáticamente hablando, y=x) donde tendría sentido que se relacionen de esta forma, ya que la cantidad de “dormitorios” debe ser menor o igual que la cantidad de “ambientes” en cada propiedad. Para estos Outliers hemos decidido trabajarlos a mano buscándolos en el dataset y analizando su descripción, que en varios ocasiones, nos brindaba información clave para esas propiedades. Por ejemplo, varias descripciones (variable nombre en nuestro dataset) mencionan la cantidad de “ambientes” de la propiedad que se está vendiendo. Y en los casos que no era así, hemos decidido imputar medianas o promedios analizando ambas variables entre sí. Es decir, si tengo una propiedad de 4 ambientes, ¿Cuál es el promedio de dormitorios en el dataset? ¿Y la mediana?. Siguiendo esta convención hemos solventado muchos Outliers (sin sentido) en esta sección, mejorando la precisión y legibilidad de los gráficos.

Para el caso de “superficie total” y “superficie cubierta”, los hemos trabajado de forma similar a los “ambientes” y “dormitorios” ya que también tienen una correlación positiva entre ellas de la misma forma. Hemos graficado tanto boxplots para ambas variables como un scatter plot (Dispersión) para observar todas las propiedades de nuestro dataset. Con ellos, a simple vista se veían claros Outliers de casos donde las “superficie totales” eran muy altas pero las “superficie cubiertas” eran muy bajas, incluso a veces, viceversa. Lo “ideal” sería que el gráfico de Dispersión se vea similar a una recta diagonal (matemáticamente hablando, y=x) donde tendría sentido que se relacionen de esta forma, ya que la cantidad de “superficie cubierta” debe ser menor o igual que la cantidad de “superficie total” en cada propiedad. Para estos Outliers hemos decidido trabajarlos a mano buscándolos en el dataset y analizando sus “ambientes” y “dormitorios”, que llegado a este punto, como ya las habíamos tratado antes, ahora nos brindan buena información para esas propiedades. Por ejemplo, varias descripciones (variable nombre en nuestro dataset) mencionan la cantidad de “ambientes” de la propiedad que se está vendiendo. Y en los casos que no era así, hemos decidido imputar medianas o promedios de superficies analizando ambas variables entre sí y también los “ambientes” y “dormitorios”. Es decir, si tengo una propiedad de 4 ambientes, ¿Cuál es el promedio de “superficie total” en el dataset? ¿Y la mediana? O por otra parte, si tengo una propiedad de 4 ambientes, ¿Cuál es el promedio de “superficie cubierta” en el dataset? ¿Y la mediana?. Siguiendo esta convención hemos solventado muchos Outliers (sin sentido) en esta sección, mejorando la precisión y legibilidad de los gráficos.

Luego, en el caso de “precio” decidimos trabajar la detección de Outliers con un método univariado, ya que los gráficos que hemos realizado de esta columna, son casi ilegibles por la cantidad de Outliers tan extremos en él. Por ejemplo, hay una propiedad en “Caballito” de 2 “ambientes” que su “precio” es de 21 millones de USD. Por casos así, los gráficos, como

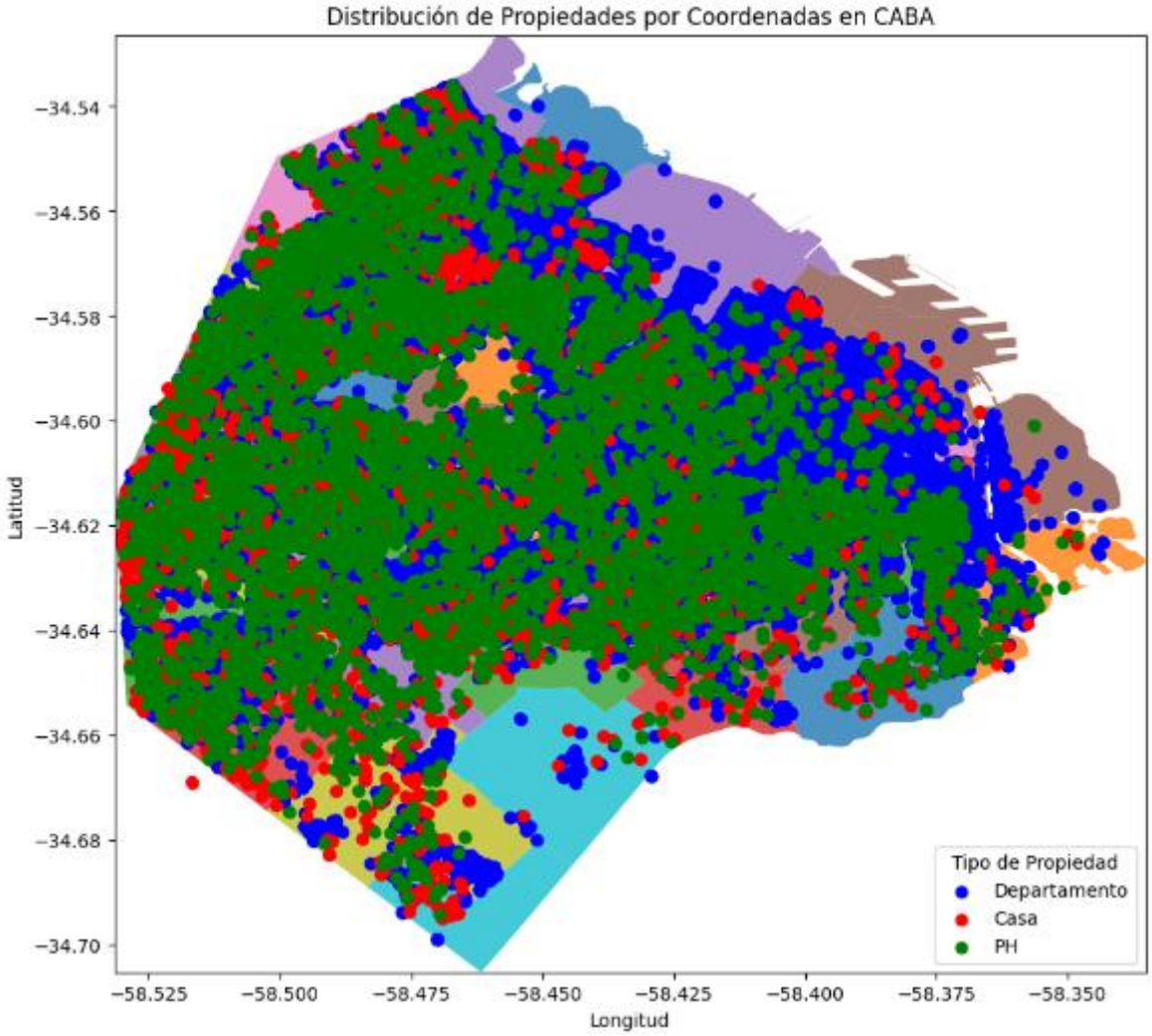
el boxplot, dejan mucho que desear visualmente hablando y hay un enmascaramiento de otros posibles Outliers que no se ven debido al desplazamiento que generan esos casos extremos. Por lo tanto, utilizamos el método “Z-Score Modificado” para detectar todos los Outliers mediante su métrica y su regla de oro (umbral en 3,5). De esta forma, según este método, hay aproximadamente 6800 Outliers. La decisión que hemos tomado para ellos ha sido, analizando las variables “ambientes”, “tipo” y “superficie total” de cada propiedad, calcular un promedio de precios para imputar en cada caso que sea Outliers. Mismo algoritmo que realizamos para Superficies. Es decir, si tengo un departamento de 4 ambientes con 170 mts^2, ¿Cuál es el promedio de precios con esas características en el dataset?

Y por último, como es solicitado, también hemos trabajado la detección de Outliers de “precio” de forma multivariada con “superficie total”, mediante el método “Distancia de Mahalanobis”. De todas formas no hemos llegado a terminar de cerrar esta parte y tomar la decisión sobre alguna convención a tomar. Es lo único faltante de este punto.

Visualizaciones:

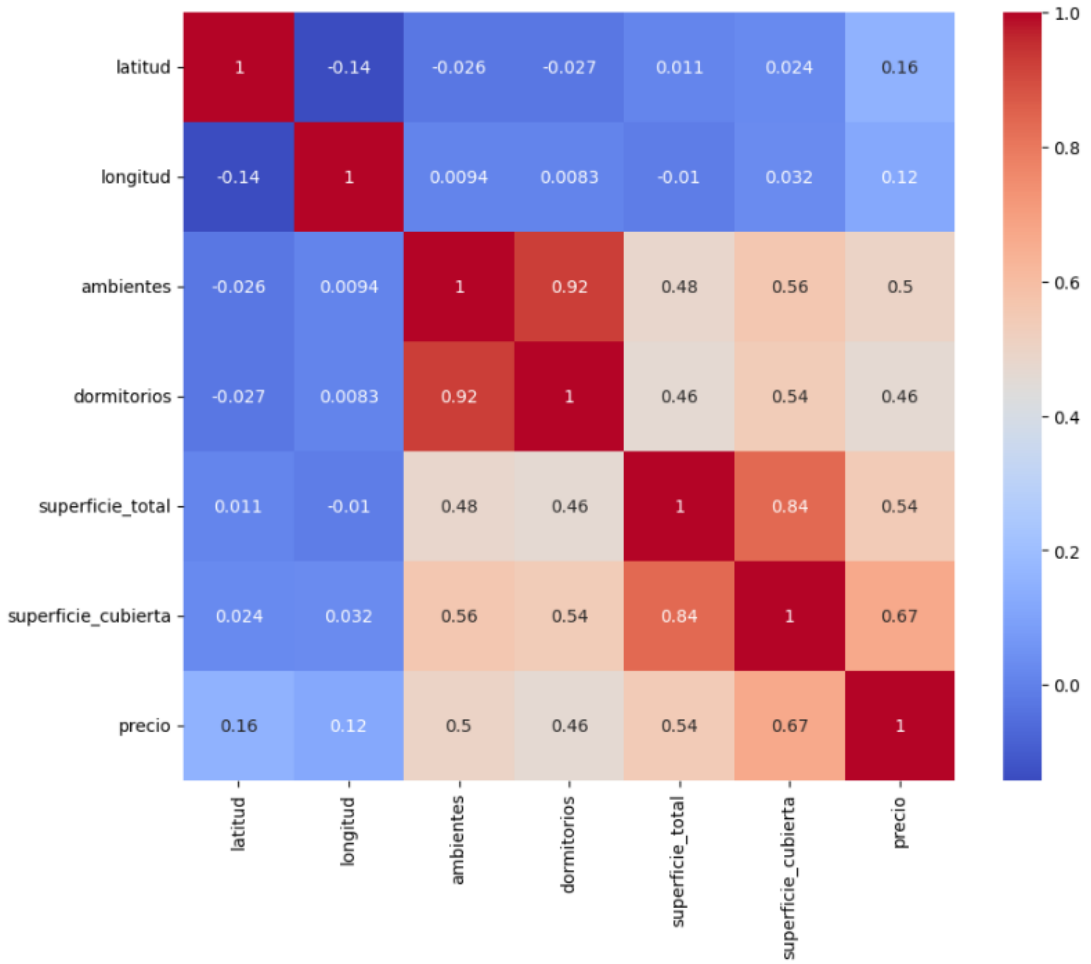
Como primera visualización gráfica a traer al informe, elegimos el gráfico de puntos en CABA, luego de ser depurado y con los Outliers de coordenadas geográficas corregidas. Este gráfico muestra la relación del “barrio” con “latitud” y “longitud” mediante los puntos azules. Como observamos, la visualización es coherente ya que ningún punto está por fuera de CABA, cosa que en una versión anterior de este mismo gráfico sí sucedía. También es muy notable la numerosa cantidad de propiedades en el dataset debido a la superposición de todos los puntos en el mapa.

Como nueva adquisición, hemos incluido una clasificación de las propiedades sobre su tipo, incluida una leyenda, para que sean un poquito más visibles todas ellas, además de ver a qué grupo pertenecen.



Como segunda visualización gráfica a traer al informe, elegimos el Heatmap, el gráfico de correlaciones lineales entre las variables cuantitativas mediante la correlación de Pearson. Sin embargo, esta es una nueva versión más actualizada que la anterior mostrada, ya que este último lo hemos hecho luego de analizar y tratar todos los datos faltantes y Outliers. Como podemos observar, en general, las correlaciones han mejorado entre todas. Antes habíamos mencionado que sonaría “lógico” o uno supondría que la “superficie total” y “superficie cubierta” estarían correlacionadas a “ambientes” y “dormitorios” pero NO sucedió. En cambio, habiendo depurado el dataset de errores, nulos y outliers, ahora se vé que las correlaciones entre estas cuatro columnas del dataset, han mejorado bastante; incluso hasta aproximarse a 0,50. También hablando de a pares, las correlaciones entre “ambientes” y “dormitorios” se han reforzado llegando hasta 0,92. De la misma forma, el caso de “superficie total” con “superficie cubierta”, donde su correlación escaló hasta 0.84. En el caso del precio, mejoró todas sus correlaciones excepto con “latitud“ y “longitud”, por obvias

razones. Pero las demás, ahora precio si tiene un apoyo sobre las otras columnas, llegando también a valores aproximados a 0,50.



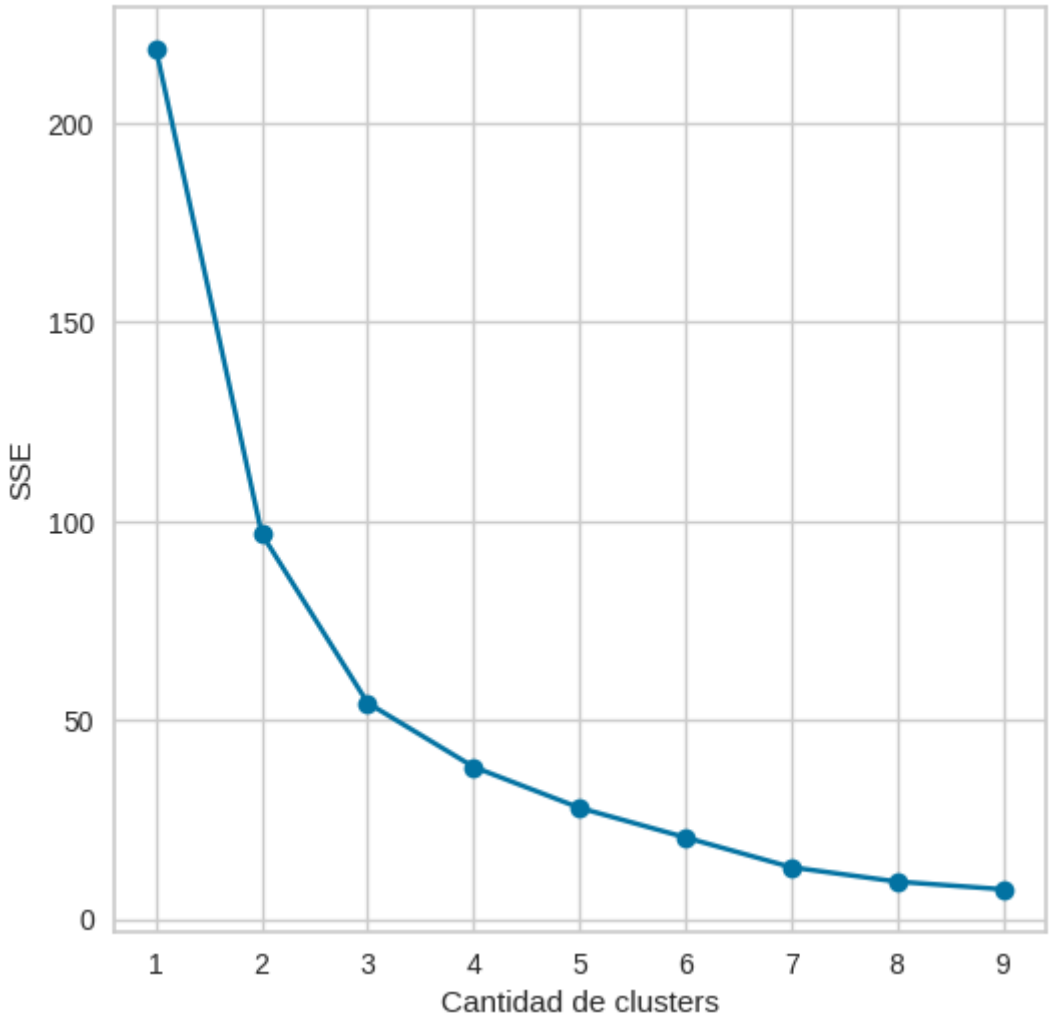
Clustering:

Para obtener la tendencia al clustering utilizamos el **método de Hopkins**. Este mismo, a través de su estadística, nos indica que un conjunto de datos es más propenso al agrupamiento o clustering si la misma da cercana a 1. En nuestro caso el resultado de la estadística de Hopkins es de **0.00034**. Del resultado podemos obtener dos conclusiones:

1. Poca tendencia al agrupamiento.
2. Observaciones distribuidas de manera uniforme y aleatoria.

Es de mencionar que para obtener la estadística utilizamos los atributos **ambientes** y **superficie_total**, ya que consideramos que son las más representativas.

Al momento de seleccionar la cantidad de grupos primero utilizamos el método del codo o **Elbow method**, para poder visualizar el número de grupos apropiado que se utilizaran en K-means posteriormente.



Como se puede observar la curva se suaviza luego del valor **3**, en el eje “Cantidad de clusters”, y nos indica que luego del codo la disminución de la inercia se estanca.

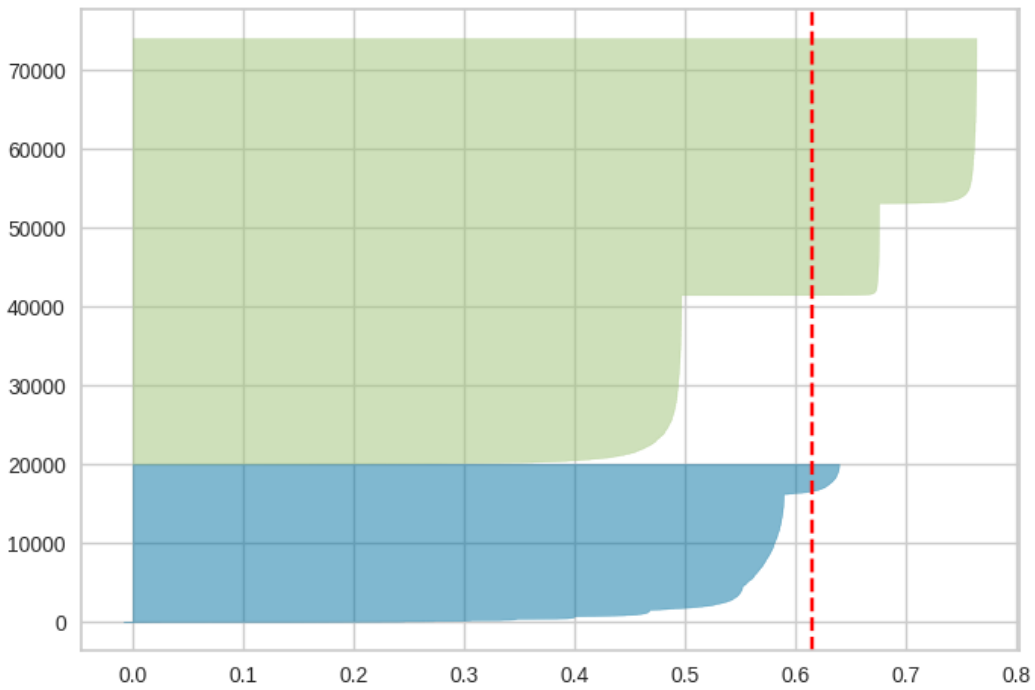
Como segundo método para determinar la cantidad apropiada de clusters utilizamos Silhouette, el cual arroja un resultado más preciso y por este motivo tendrá más peso al momento de seleccionar la cantidad.

Como primer paso buscaremos los coeficientes de Silhouette para K = i, con i ϵ [2;9], sabiendo que los mejores candidatos son los más cercanos a 1. Mientras más cerca a 1 están los coeficientes, más alejados de los clusters vecinos estará la observación y por ende, mejor clasificadas están las instancias.

K (cantidad de clusters)	Coeficiente de Silhouette
2	0.6144717514915142
3	0.6582392073264389
4	0.6942160956451406
5	0.8271630248216466
6	0.8291369627723643
7	0.8651194323960418
8	0.8658019797844206
9	0.8751094508835678

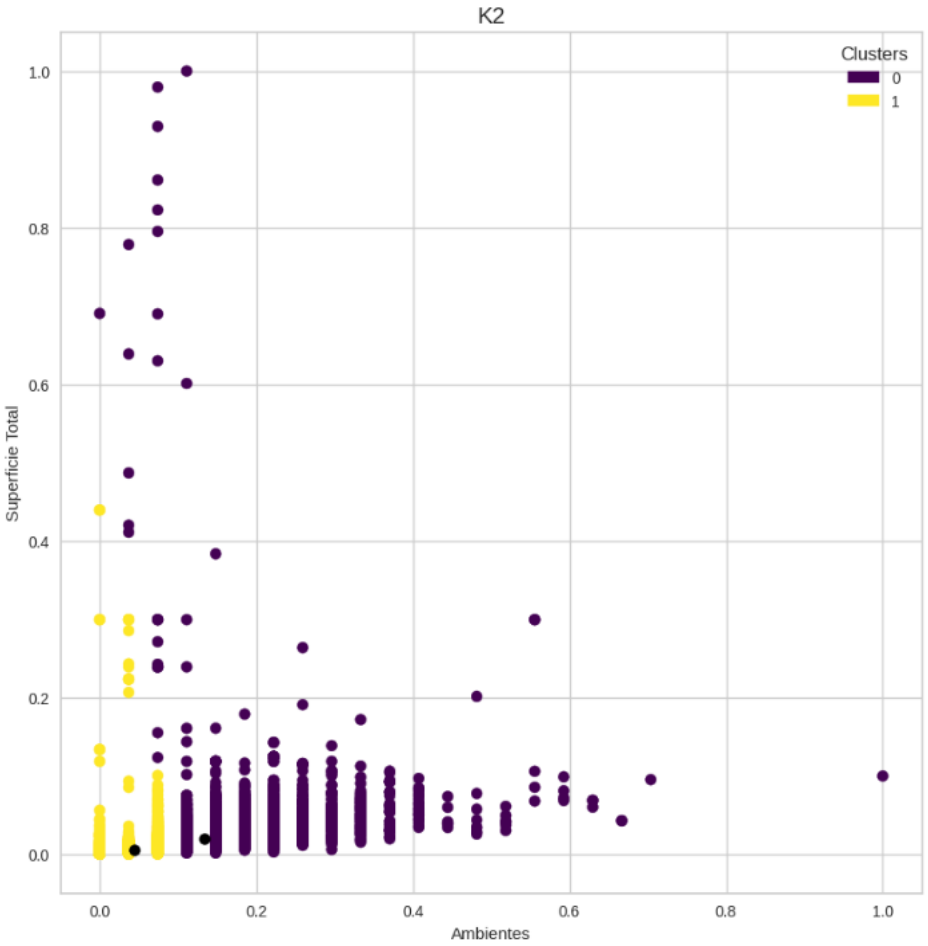
De la tabla podemos concluir que los mejores **K son 5, 6, 7, 8 y 9** pero al momento de graficar, en todos los casos, tenemos uno o más agrupaciones que no sobrepasan la media de Silhouette. Para **K = 3, 4** obtenemos el mismo caso.

Por último nos queda **K = 2**, el cual es representado por el siguiente gráfico.



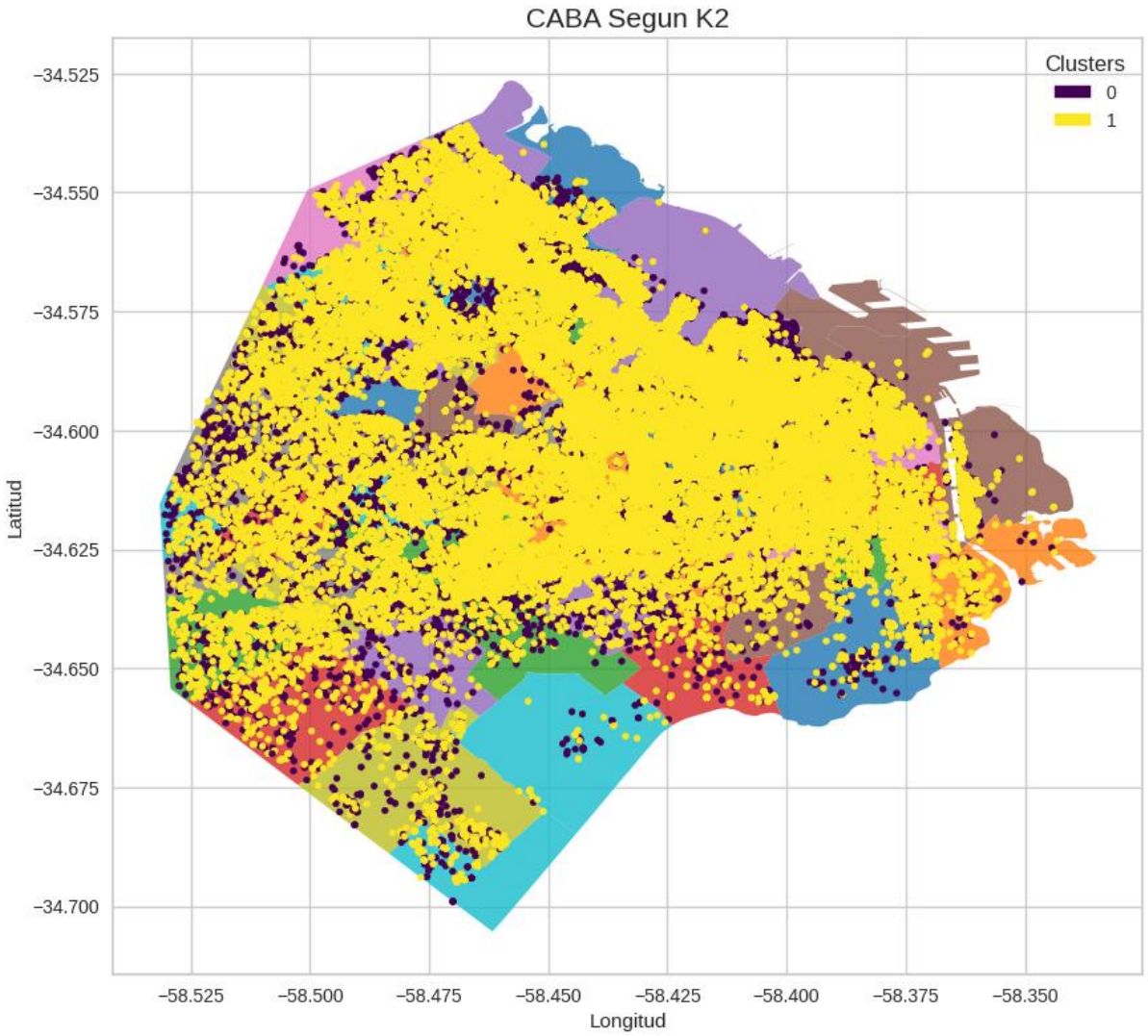
Si bien el tamaño de las dos agrupaciones no es homogéneo, ambas sobrepasan la media y por lo tanto consideramos que **es el mejor K y el que vamos a utilizar para K-means**.

En conclusión, teniendo en cuenta la precisión del coeficiente de Silhouette por sobre el método del codo, y analizando los gráficos para visualizar que todas las agrupaciones sobrepasan la media, decidimos utilizar K igual a 2 como mejor hiperparámetro.



Luego, al entrenar el modelo, observando el Scatterplot podemos estimar que los clusters se agruparon en base a la cantidad de ambientes, donde a partir de un umbral de aproximadamente $ambientes > 0.1$ las instancias pertenecen al grupo 0.

Por último, la distribución de las propiedades agrupadas en un mapa de BS. AS. se ve de la siguiente manera:



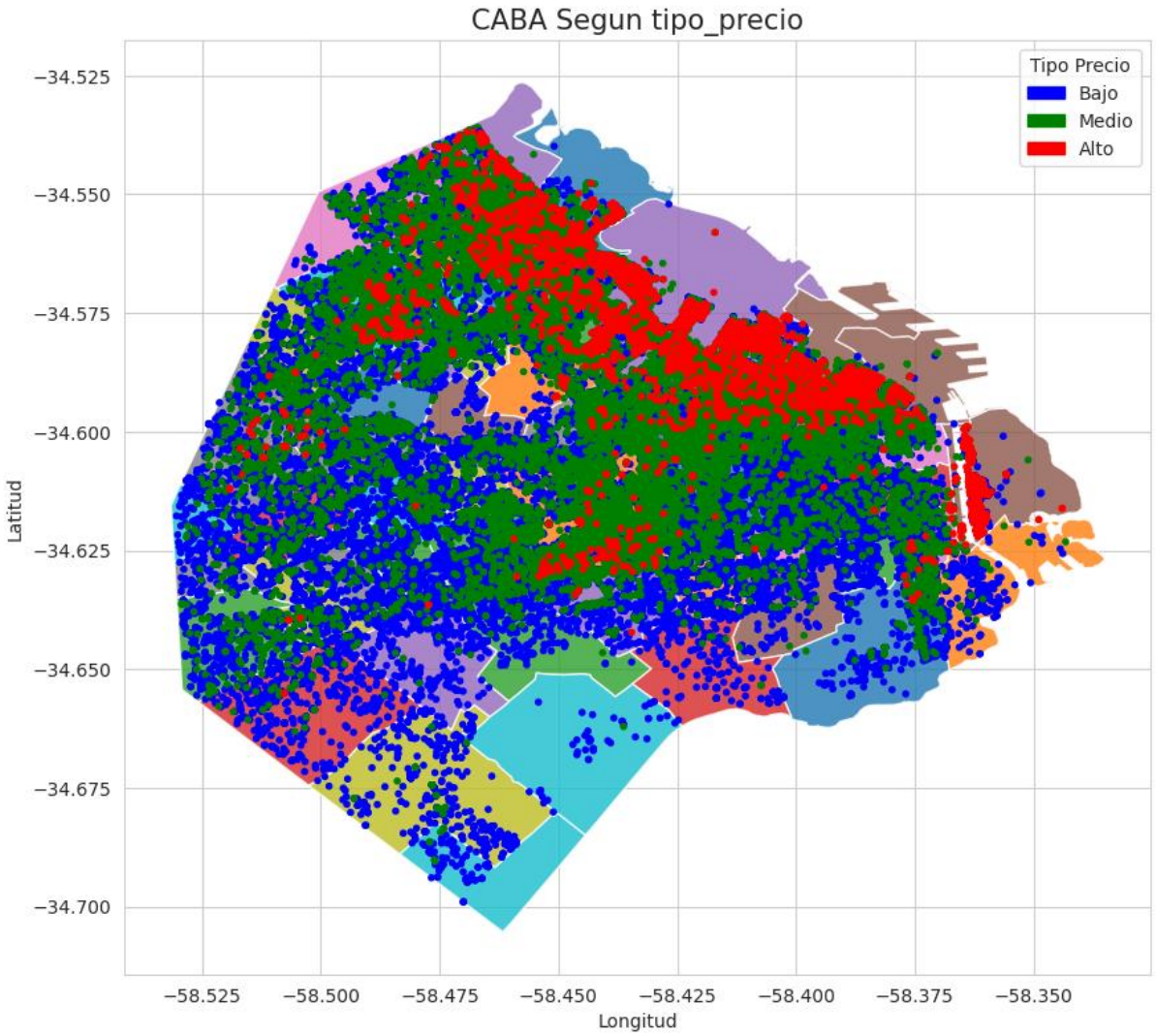
Clasificación:

La alternativa seleccionada para construir la variable “tipo_precio” fue la siguiente. Hemos trabajado la variable “pxm2” relativa a cada tipo de propiedad, y luego, la hemos dividido en tres intervalos. El primero con el 25% de las observaciones, el siguiente con el 50% y el último con el 25% de las observaciones restantes, es decir, trabajando con los cuartiles.

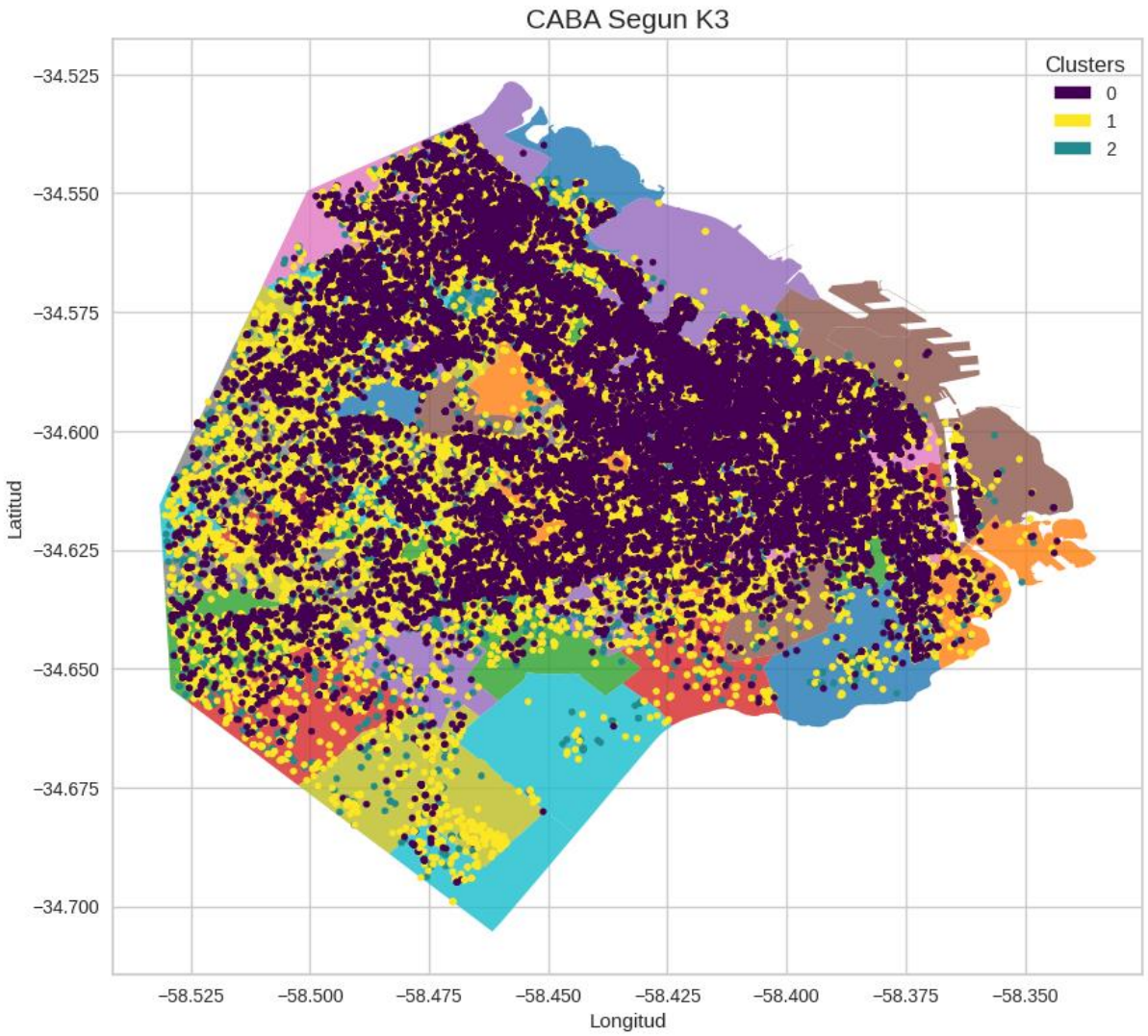
De esta forma, en la categoría tipo de precio “Bajo” nos ha quedado el primer cuartil de la cantidad de observaciones de las propiedades que son “Casa”, el primer cuartil de las propiedades que son “PH” y el primer cuartil de los “Departamentos”. A su vez, en la categoría tipo de precio “Medio” nos ha quedado el siguiente 50% de las observaciones según su “tipo” de propiedad. Y por último, en la categoría tipo de precio “Alto” nos ha quedado el 25% restante de las propiedades según su “tipo”.

Tomamos la decisión de dividir la variable “tipo_precio” de esta manera ya que es la distribución que más “se asemeja” a la realidad en nuestra perspectiva. Uno cuando busca una vivienda para comprar o simplemente averiguar precios, analiza su precio según sus características puntuales. Es decir, si voy a mirar un departamento de un ambiente situado en Puerto Madero, probablemente sea muy caro por su ubicación pero el departamento en sí, quizás no sea muy lujoso como para valer dicho precio. De la misma forma, el caso contrario. Una casa de varios ambientes ubicada en Villa Devoto, muy lujosa quizás sea muy rentable si está a un precio “bajo”. En conclusión, cuando uno analiza precios de propiedades, lo hace situacionalmente hablando. Por ende, si hubiéramos dividido la variable “tipo_precio” sin tener en cuenta los tipos de propiedad, estaríamos generalizando mucho el análisis y quizás, esos dos ejemplos recién mencionados “caerían” a la misma categoría de precio “Alto”. Sin embargo, al realizar un pequeño análisis previo de los “tipos“ de propiedad, estaríamos profundizando ese análisis y ajustándolo un poco más a la realidad.

Adjuntamos el siguiente gráfico de un mapa de CABA con los avisos coloreados y su respectiva leyenda, para observar como ha quedado la distribución de las propiedades según el agrupamiento realizado por la variable tipo_precio con el análisis seleccionado.

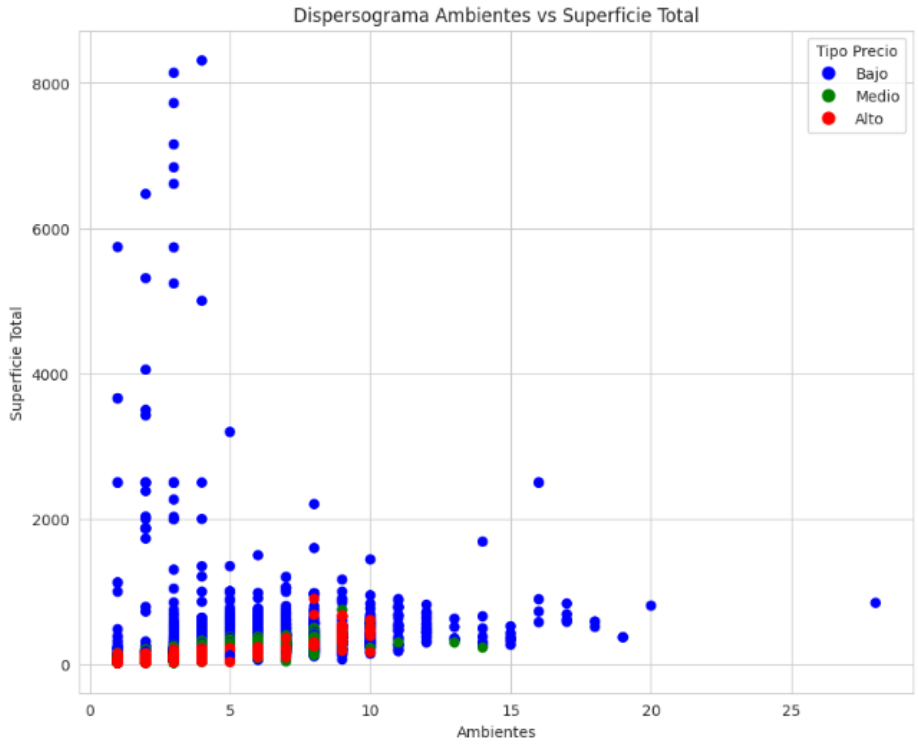


Luego, hemos realizado un análisis y intentamos encontrar algunas similitudes entre el agrupamiento de K-Means con 3 grupos y esta distribución de la variable ”tipo_precio”; llegando a la siguiente conclusión. Por cómo se distribuye K Means = 3 en el mapa de CABA, podemos inferir que la mayoría de propiedades con tipo_precio “Alto”, forman parte del cluster 0 (amarillo) o del cluster 1 (violeta). Esto puede llegar a ser intuitivo ya que estas propiedades son la minoría dentro de la variable, y esos dos clusters son la mayoría. Adjuntamos el siguiente gráfico de un mapa de CABA con los avisos coloreados y su respectiva leyenda, para observar como ha quedado la distribución de las propiedades según el agrupamiento realizado por el K-Means (con k=3).

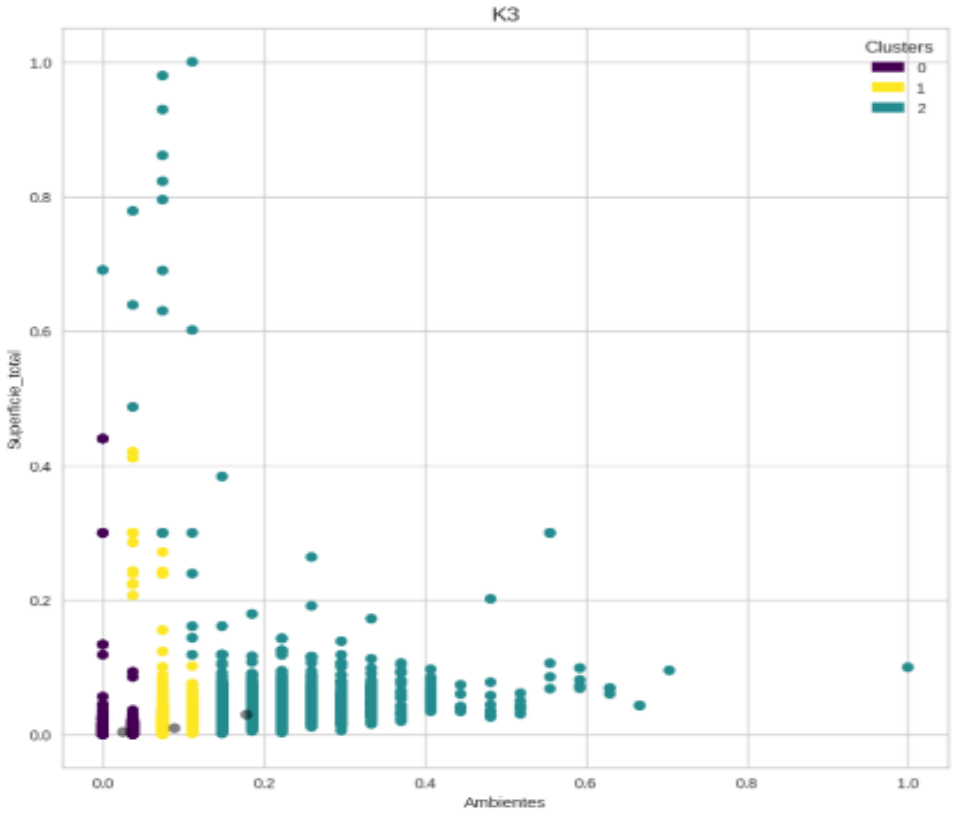


Ahora, trataremos de hallar alguna similitud viendo cómo se distribuyen ambas clasificaciones en un ScatterPlot de ejes: ambientes y superficie_total, debido a que son las variables que forman los clusters en K-means. Sin embargo, no hemos podido concluir en ninguna relación entre los clusters formados y la clasificación obtenida por la variable “tipo_precio”

Adjuntamos el siguiente gráfico de un mapa de CABA con los avisos coloreados y su respectiva leyenda, para observar como ha quedado la distribución de las propiedades según el agrupamiento realizado por el K-Means (con k=3).



Y, para que se observe que no tienen algún patrón de similitudes. Adjuntamos el siguiente gráfico de dispersión con los avisos coloreados y su respectiva leyenda, para observar como ha quedado la distribución de las propiedades según el agrupamiento realizado por el K-Means (con k=3).

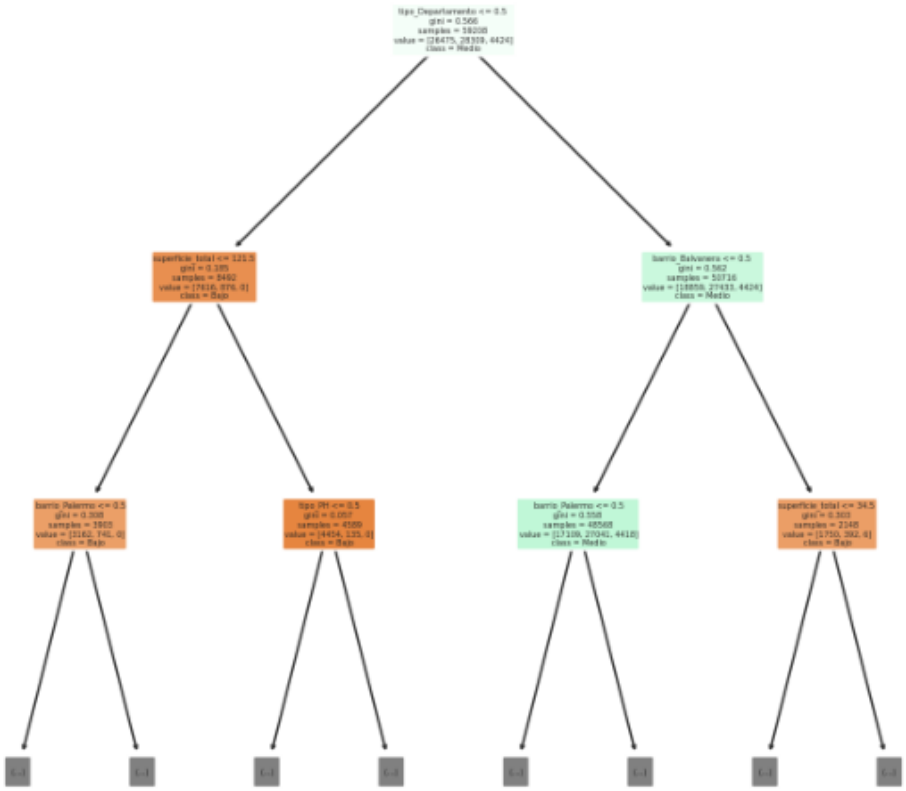


a) Construcción de Modelos:

Árbol de Decisión:

Primero hemos realizado un árbol de decisión básico sin ningún hiperparámetro optimizado, incluso 100% default. Sus números han sido bastante correctos, las cuatro métricas rondan casi por 0,74 aproximadamente. Para ser un árbol sin ninguna optimización, son números bastante acertados. Luego, hemos decidido jugar y buscar optimizar los hiperparámetros mediante Randomized Search Cross Validation. En primera instancia, utilizamos la métrica "F1 Score" para optimizar los hiperparámetros, ya que era la métrica más bajita (aunque por poco, ya que era de 0,73). Luego, seleccionamos un k = 5 folds pero eran pocos y los árboles generados tenían un escaso score, rondando por los 0,54, aproximadamente. Así que decidimos aumentar a un k = 10 folds y ya los score aumentaban hacia los 0,66. Hasta que decidimos dejarlo en k = 20 folds y que pruebe 40 combinaciones, es decir, llegaría a probar unos 800 árboles.

Sin embargo, fue muy difícil hasta casi improbable, mejorar las métricas generadas por el modelo creado de forma default. Hemos logrado encontrar un combinación de hiperparametros, los cuales casi ni modifican las métricas (lo reducen pero en una medida ínfima, dejándolas en 0,72). De esta manera, hemos llegado a la conclusión de que (como se nos ha mencionado en las clases teóricas) y por estar trabajado con big data, los árboles no son tan buenos predictores como parecían, siendo el árbol 100% default uno de los mejores. Por ende, será el seleccionado para poner sus métricas en el cuadro de resultados. De paso, adjuntamos un gráfico representativo del árbol ya que, si quisiéramos graficar el árbol completo, al ser big data, es súper extenso y casi ilegible.



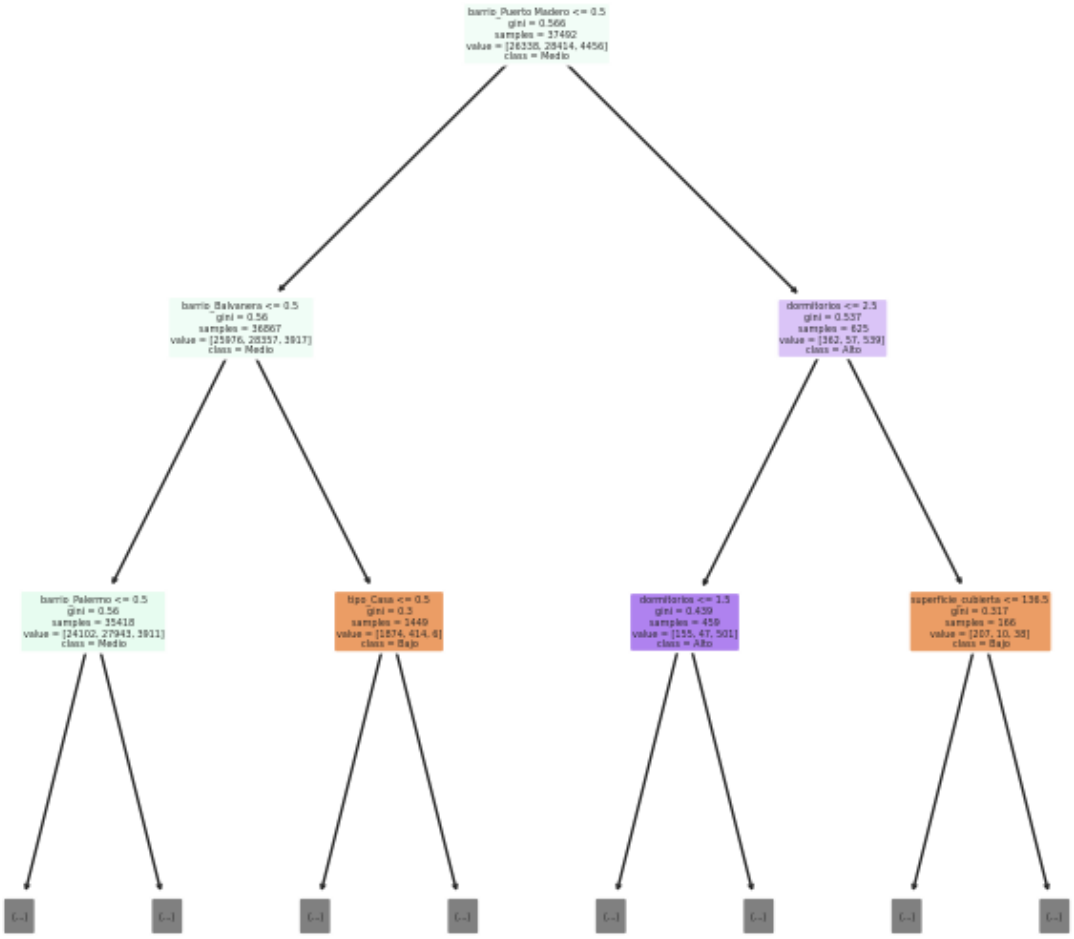
Random Forest:

Primero hemos realizado un random forest arbitrario sin ningún hiperparámetro optimizado, tratando de crearlo, lo más parecido al default posible (los hiperparámetros que se han elegido han sido el max features = “auto”, que se cambió el default recientemente; el random_state para eliminar la aleatoriedad entre ejecuciones, etc). Sus números han sido bastante correctos, las cuatro métricas rondan por 0,751 aproximadamente. Para ser un bosque sin ninguna optimización, son números bastante acertados e incluso un poquito mejores que los del mejor árbol de decisión.

Luego, hemos decidido jugar y buscar optimizar los hiperparámetros mediante Grid Search Cross Validation. En primera instancia, utilizamos la métrica “F1 Score” para optimizar los hiperparámetros, ya que era la métrica más bajita (aunque por muy poco, ya que era de 0,752). Con esta situación, con los hiperparametros recomendados, su Score juntos al de las demás métricas aumentaban hacia los casi 0,758 aproximadamente. A priori no parecería ser muy grande pero era un aumento. Luego, realizamos un Grid Search CV pero seleccionando más de una métrica a mejorar; las cuales fueron Accuracy, F1 Score. Este nuevo modelo nos devolvió un ínfimo peor resultado que al seleccionar sólo F1 Score como

la métrica a mejorar. Sin embargo, seguía mejorando las métricas en general, con respecto al bosque por default, y llevándolas hacia los 0.755, aproximadamente.

De esta manera, hemos llegado a la conclusión de que como los bosques utilizan muchos árboles en paralelo y al estar trabajando con big data, también los Random Forest no son tan buenos predictores para clasificar, en este caso. Mencionando que el mejor bosque supera las métricas del mejor árbol, en una baja medida; el cual será el seleccionado para poner sus métricas en el cuadro de resultados. Cómo Random Forest es un ensamble de árboles trabajados en paralelo, adjuntamos solo un gráfico de unos de esos 50 estimadores, específicamente el árbol num 7 del bosque como representante.



Modelo a Elección (XGBoost para Clasificación):

Primero hemos realizado un XGBoost arbitrario sin ningún hiperparámetro optimizado, tratando de crearlo, lo más parecido al default posible (salvo el random_state para eliminar la aleatoriedad entre ejecuciones, etc). Sus números han sido bastante correctos, las cuatro métricas rondan por 0,746 aproximadamente. Para ser un modelo sin ninguna optimización, son números bastante acertados e incluso un poco mejores que los del mejor árbol de decisión, aunque un poco peor que el mejor bosque.

Luego, hemos decidido jugar y buscar optimizar los hiperparámetros mediante Grid Search Cross Validation. En primera instancia, utilizamos la métrica “F1 Score” para optimizar los hiperparámetros, ya que era la métrica más bajita (aunque por muy poco, ya que era de 0,734 aprox). Con esta situación, con los hiperparametros recomendados, su Score juntos al de las demás métricas aumentaban hacia los casi 0,76 aproximadamente. A priori no parecería ser muy grande pero era un aumento, el cuál posiciona a este modelo como, hasta ahora, el mejor predictor de clasificación en nuestro tp. Luego, realizamos un Grid Search CV pero seleccionando más de una métrica a mejorar; las cuales fueron Accuracy, F1 Score. Este nuevo modelo nos devolvió, casualmente, la misma combinación de hiperparámetros que el antes mencionado. Es decir, las métricas terminan siendo las mismas y por obvias cuestiones, termina siendo el mejor modelo y el seleccionado para el cuadro de resultados.

b) Cuadro de Resultados:

Modelo	F1 - Test	Precision Test	Recall Test	Accuracy Test
Árbol de Decisión	0,7325	0,7315	0,7345	0,7345
Random Forest	0,7566	0,7576	0,7597	0,7597
XGBoost	0,7580	0,7626	0,7631	0,7631

Árbol de Decisión: (Creado 100% Default) => {'max_depth': None, 'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 2, 'ccp_alpha': 0,0}

Random Forest: {'criterion': 'entropy', 'min_samples_leaf': 1, 'min_samples_split': 6, 'n_estimators': 50}

XGBoost: {'learning_rate': 0.3, 'n_estimators': 100, 'max_depth': 15, 'reg_lambda: 2}

Regresión:

a) Construcción de Modelos:

En principio, se buscaron los features que tiendan a una relación lineal con Precio, a través de un pairplot y lo verificamos con la correlación. Del gráfico nos quedamos con ambientes, dormitorios, superficie cubierta y superficie total.

Sabemos que las regresiones son susceptibles a los outliers. Por este motivo procederemos a eliminar estos mismos con la regla del 1.5*IQR, dejando las observaciones que están entre 1.5 veces el rango entre cuartiles.

Una vez hecho creando y entrenando los modelos, evaluaremos las siguientes métricas:

- Error cuadrático medio (MSE)
- Raíz del error cuadrático medio (RMSE)
- Error absoluto medio (MAE)
- Coeficiente de Determinación (R2 Score)

KNN:

En primer lugar, hemos realizado un K-nearest neighbors arbitrario sin ningún hiperparámetro optimizado, incluso creándolo 100% default. Sus números han sido intuitivamente bajos, los errores son muy grandes en magnitud y el Coeficiente de Determinación (R2 Score) tiene un valor de 0,606. Para ser un KNN default, quizás, son números acertados ya que la magnitud de los errores es inmensa al estar trabajando precios de propiedades en dólares. Estamos hablando de un RMSE de 67 mil o un MSE de 4500 millones.

Luego, hemos decidido jugar y buscar optimizar los hiperparámetros mediante Grid Search Cross Validation con 50 folds. En primera instancia, utilizamos la métrica “R2 Score” para optimizar los hiperparámetros, ya que la métrica es una “especie” de performance general del modelo. Con esta situación, con los hiperparametros recomendados, su Score juntos al de los errores, no llegaban a mejorar el default, incluso empeoraron en una ínfima medida. Luego, realizamos un Grid Search CV pero seleccionando el “MSE” como la métrica a mejorar, ya que era el valor más alto. Este nuevo modelo nos devolvió un mejor resultado que el default y que seleccionando el R2 Score como la métrica a mejorar. Por ende, mejoraron todas las métricas en general, llevando el R2 Score hacia los 0.615, aproximadamente; y reduciendo los errores en una gran medida. Éstos seguían teniendo

una magnitud enorme (observar el cuadro de resultados), pero se redujeron en grandes cantidades.

Sin embargo, no conformes con esta mejora, decidimos escalar los datos ya que KNN se beneficia de esta técnica. Dicho y hecho, realizamos un escalado PowerTransformer en los datos. Luego, hemos realizado un KNN arbitrario sin ningún hiperparámetro optimizado, 100% default pero con el escalado hecho. Sus números han sido bastantes mejores que el KNN optimizado antes mencionado, con un Coeficiente de Determinación (R2 Score) de valor 0,641 aprox. Por ende, hemos decidido jugar y buscar optimizar los hiperparámetros mediante Grid Search Cross Validation con 50 folds, seleccionando el MSE como la métrica a mejorar. Este nuevo modelo nos devolvió un mejor resultado que todos los anteriores, aumentando aún más las métricas en general, llevando el R2 Score hacia los 0.676, aproximadamente; y reduciendo los errores en una gran medida. Éstos seguían teniendo una magnitud enorme (observar el cuadro de resultados), pero se redujeron en grandes cantidades. Estamos hablando de un RMSE de 66 mil o un MSE de 4300 millones. En comparación a los anteriores, han disminuido bastante.

XGBoost:

En primer lugar, hemos realizado un XGBoost arbitrario sin ningún hiperparámetro optimizado, creándolo, lo más parecido al default posible (salvo el random_state para eliminar la aleatoriedad entre ejecuciones, etc). Sus números también han sido intuitivamente bajos, sin embargo, mejor que los del KNN. Los errores siguen siendo muy grandes en magnitud y el Coeficiente de Determinación (R2 Score) tiene un valor de 0,638 aprox. Para ser un XGBoost default, son números acertados ya que como mencionamos anteriormente, la magnitud de los errores es inmensa al estar trabajando precios de propiedades en dólares. Estamos hablando de un RMSE de 64 mil o un MSE de 4100 millones. En comparación al KNN, se ha reducido bastante el MSE, como beneficio. Sin embargo, ha aumentado un poquito el MAE y disminuido el R2 Score, como perjudicial.

Luego, hemos decidido jugar y buscar optimizar los hiperparámetros mediante Grid Search Cross Validation con 10 folds. En primera instancia, también utilizamos la métrica “R2 Score” para optimizar los hiperparámetros. Con esta situación, con los hiperparametros recomendados, su Score juntos al de los errores, no llegaban a mejorar el default, incluso empeoraron. Luego, realizamos un Grid Search CV pero seleccionando el “MSE” como la métrica a mejorar, por ser el valor más alto. Este nuevo modelo nos devolvió un ínfimo peor resultado que el default; aunque mejor que seleccionando el R2 Score como la métrica a

mejorar. Por ende, empeoran todas las métricas en general, llevando el R2 Score hacia los 0.6368, aproximadamente; y aumentando los errores, también en una ínfima medida, excepto el MAE que lo mejoraba simplemente en unidades. Concluyendo, el mejor XGBoost para Regresión, terminó siendo el 100% default con las siguientes métricas volcadas en el cuadro de resultados.

Modelo a Elección (Ada Boost):

En primer lugar, hemos realizado un Ada Boost arbitrario sin ningún hiperparámetro optimizado, creándolo, lo más parecido al default posible (salvo el random_state para eliminar la aleatoriedad entre ejecuciones, etc). Sus números han sido, intuitivamente, los más bajos en comparación a los otros dos modelos. Esto se debe a que el Ada Boost es un modelo que tarda menos en entrenarse y suele trabajar mejor con datasets que tengan mucho ruido. Todo esto, a costa de resignar una mejor performance a comparación del Gradient Boosting, por ejemplo. Los errores siguen siendo muy grandes en magnitud, incluso los más grandes, y el Coeficiente de Determinación (R2 Score) tiene un valor de 0,514 aprox. Siendo un Ada Boost default, sus números no son tan acertados como si quizás los de los otros modelos. Estamos hablando de un RMSE de 74 mil o un MSE de 5500 millones. En comparación al KNN y al XGBoost, los errores se han incrementado bastante.

Luego, hemos decidido jugar y buscar optimizar los hiperparámetros mediante Grid Search Cross Validation con 10 folds. Ya en esta instancia, comenzamos a utilizar la métrica “MSE” como la métrica a mejorar, ya que era el valor más alto. Este nuevo modelo nos devolvió un mejor resultado que el default y que seleccionando el R2 Score como la métrica a mejorar. En consecuencia, mejoraron todas las métricas en general, llevando el R2 Score hacia los 0.522, aproximadamente; y reduciendo los errores en una gran medida. Éstos seguían teniendo una magnitud enorme (observar el cuadro de resultados), pero se redujeron en grandes cantidades.

b) Cuadro de Resultados:

Modelo	MSE	RMSE	MAE	R2 Score
KNN	4339234121	65872	39499	0,676
XGBoost	4132423645	64283	40841	0,638

Ada Boost	5454426272	73854	51225	0,522

KNN (con Escalamiento): {'metric': euclidean', 'n_neighbors': 25, 'weights': 'uniform'}

XGBoost: (Creado 100% Default) => {'learning_rate': 0.3, 'n_estimators': 100, max_depth': None}

Ada Boost: {'learning_rate': 0.2, 'n_estimators': 100, 'loss': 'linear'}

Estado de Avance:

1) Análisis Exploratorio y Preprocesamiento de Datos:

Porcentaje de Avance: 100%/100%
Tareas en curso:
Tareas planificadas:
Impedimentos:

2) Agrupamiento:

Porcentaje de Avance: 100%/100%
Tareas en curso:
Tareas planificadas:
Impedimentos:

3) Clasificación:

Porcentaje de Avance: 100%/100%
Tareas en curso:
Tareas planificadas:
Impedimentos:

4) Regresión:

Porcentaje de Avance: 100%/100%
Tareas en curso:
Tareas planificadas:
Impedimentos:

Tiempo Dedicado:

Integrante	Tarea	Prom Hs. Semana
Matias Agustin Ferrero Cipolla	Clustering Clasificación	8 hs
Franco Ricciardo Calderaro	Clasificación Regresión Armado de Reporte	8 hs
Carlos Alejandro Orqueda	Clustering Regresión Armado de Reporte	7 hs
Sebastian Kraglievich	Clustering Regresión	6 hs