Checkpoint 1 - Grupo 04

Análisis Exploratorio

El dataset con el que vamos a trabajar es de la compañía Properati, una empresa inmobiliaria fundada en Argentina. Utilizaremos los datos públicos que la empresa brinda en cuanto a ventas de inmuebles, más específicamente en Argentina durante el año 2021. Inicialmente el dataset cuenta con 20 columnas y 460154.

El primer filtrado de información lo realizaremos sobre el DS (dataset) en su totalidad, ya que solo nos interesan, para la finalidad de este trabajo, las propiedades que entren dentro de la categoría (departamento, PH o Casa), que se encuentren dentro de la Capital Federal, con un precio en dolares y que estén catalogados como ventas.

Los features más destacables pueden ser, de nuevo, por la importancia que presentan para nuestro análisis, el **tipo de propiedad**, el **place_l2** (representa si la propiedad está en capital federal o no para nosotros), el **tipo de operación** y por último el **property_currency** que nos va a dar la información sobre si la propiedad se vende en dolares o en pesos. A esta altura del análisis de los datos no realizamos ninguna suposición, simplemente hicimos un filtrado inicial de los datos.

Preprocesamiento de Datos

Detallar las tareas más importantes que realizaron sobre el dataset, les dejamos algunas preguntas cómo guía:

Se eliminaron las columnas '**place_l2**', '**operation**' y '**property_currency**' porque al haber hecho una filtración inicial de propiedades estas quedaron con la misma información en todas las filas por lo que era información redundante.

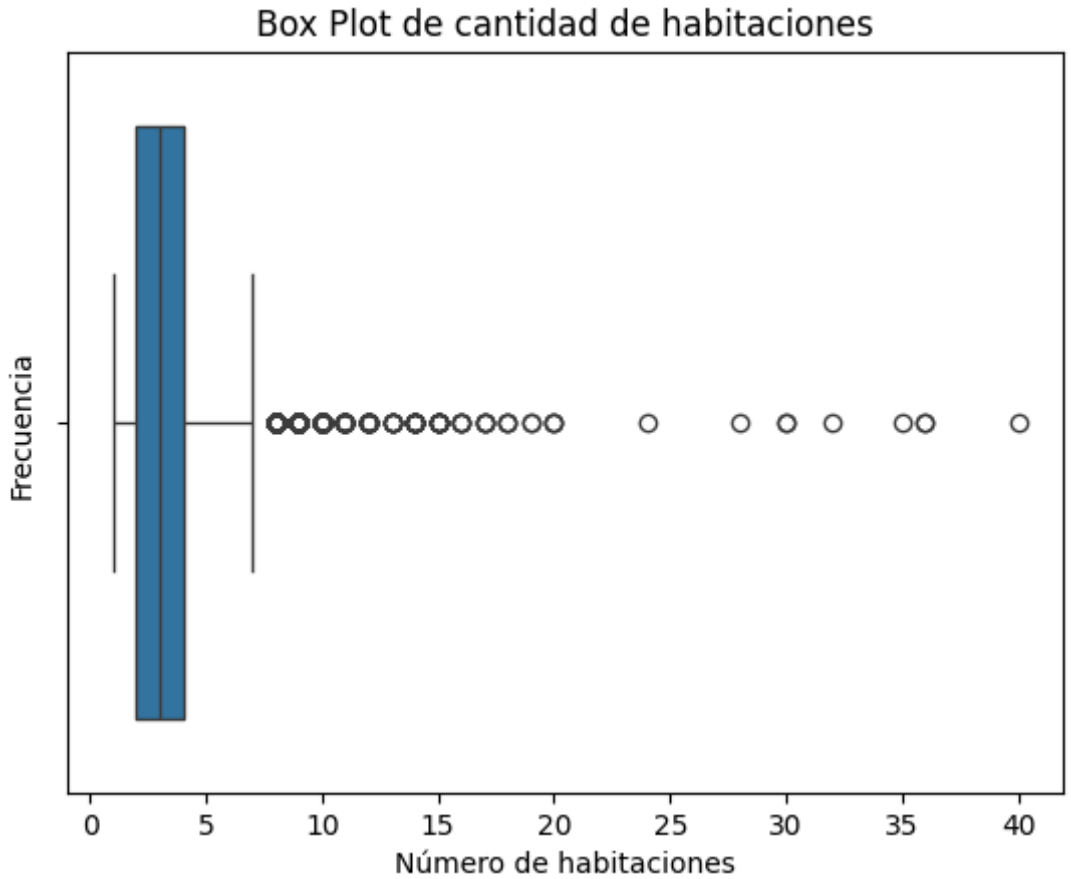
Las correlaciones más interesantes que encontramos fueron:

- '**property_price**' y '**property_surface_total**' con un coeficiente de **0.08523491719899819**
- '**property_price**' y '**property_rooms**' con un coeficiente de **0.488934080178357**
- '**property_price**' y '**property_surface_covered**'' con un coeficiente de **0.05623785863824733**
-

Si bien estas correlaciones podrían indicar una falta de relación entre estas variables, creemos que si hay una relación, y por ende suponemos que es debido a los outliers de los diferentes coeficientes. Una vez hecho el tratado de outliers, volveremos a calcular estas correlaciones.

No generamos nuevos features con respecto al DS, al menos no encontramos qué features nuevos inicializar.

Encontramos valores atípicos tanto para variables univariadas como multivariadas. En el caso de las univariadas, con un gráfico de boxplot podemos observar algunos valores atípicos para la columna de 'property_rooms:

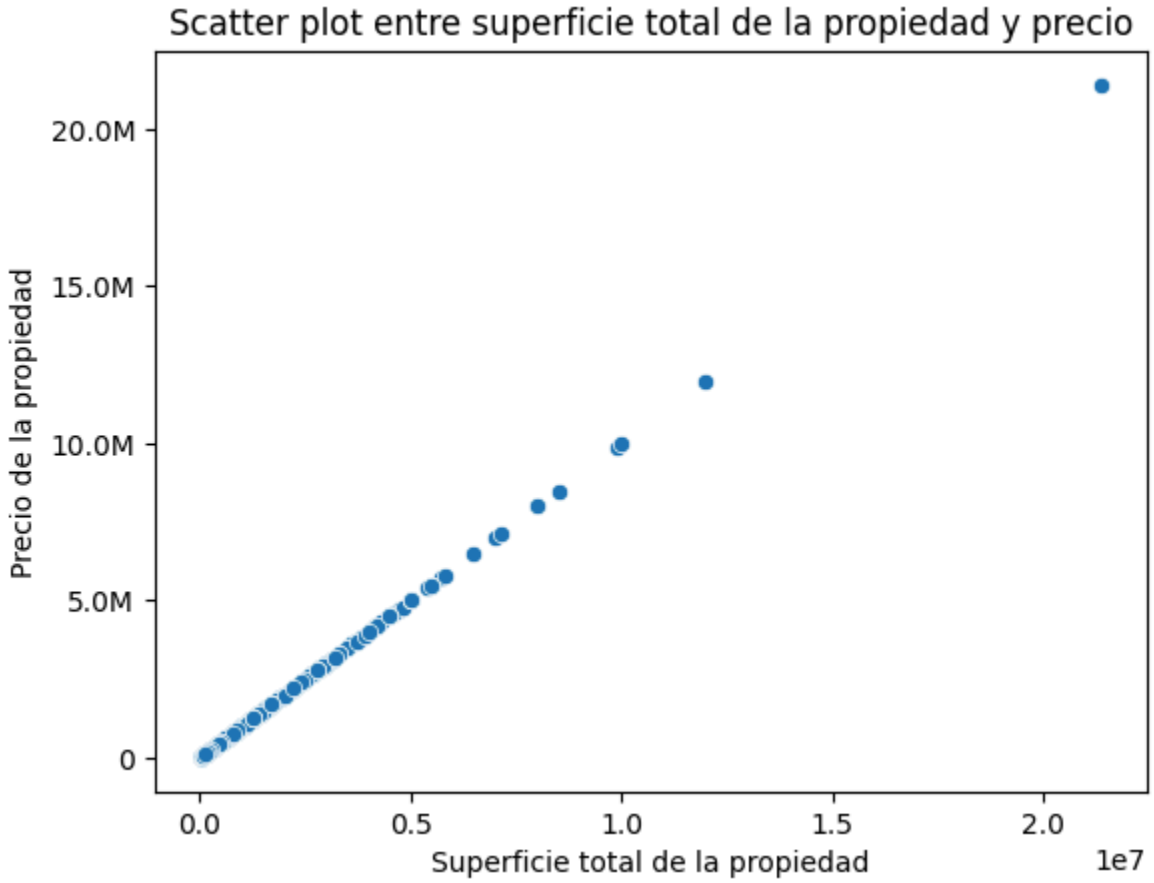


También utilizamos el método Z-score para detectar outliers en las columnas “property_rooms” y “property_bedrooms”. En ambos casos se probó con distintos valores para el umbral, hasta llegar a uno que nos devolvió valores realmente alejados de la media y que no son coherentes con lo que representan.

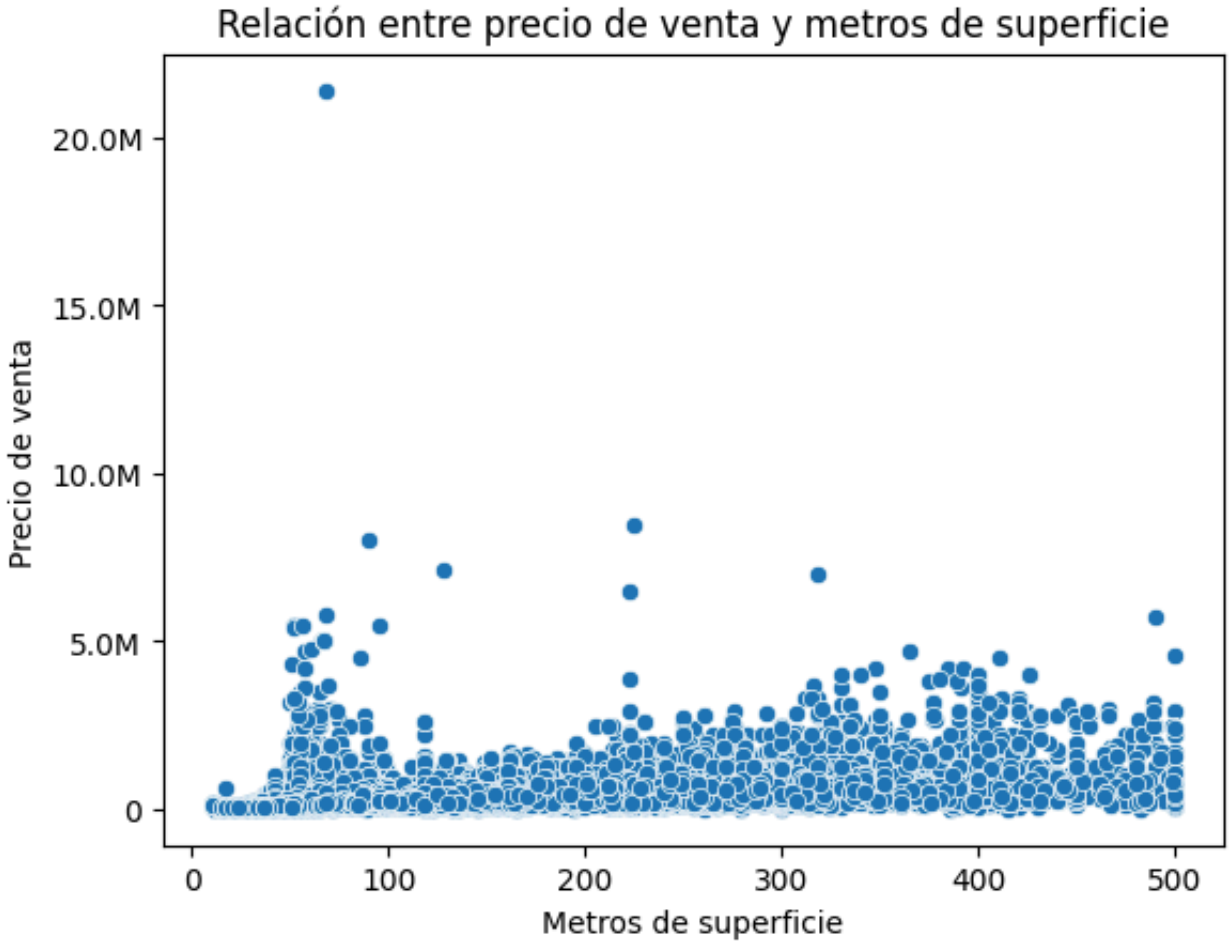
En el caso de la columna “property_rooms” se utilizó un umbral de 9, que nos devolvió 22 valores que rondan entre 16 y 40. Los cuales son demasiado altos para representar la cantidad de habitaciones en una propiedad.

En el caso de la columna “property_bedrooms” se utilizó un umbral de 5, que nos devolvió 224 valores que rondan entre 9 y 39. En este caso nos devolvió una cantidad de valores mayor que en la columna anterior y con un rango mayor. Pero consideramos que estos valores son atípicos ya que es raro encontrar propiedades con 9 o más dormitorios.

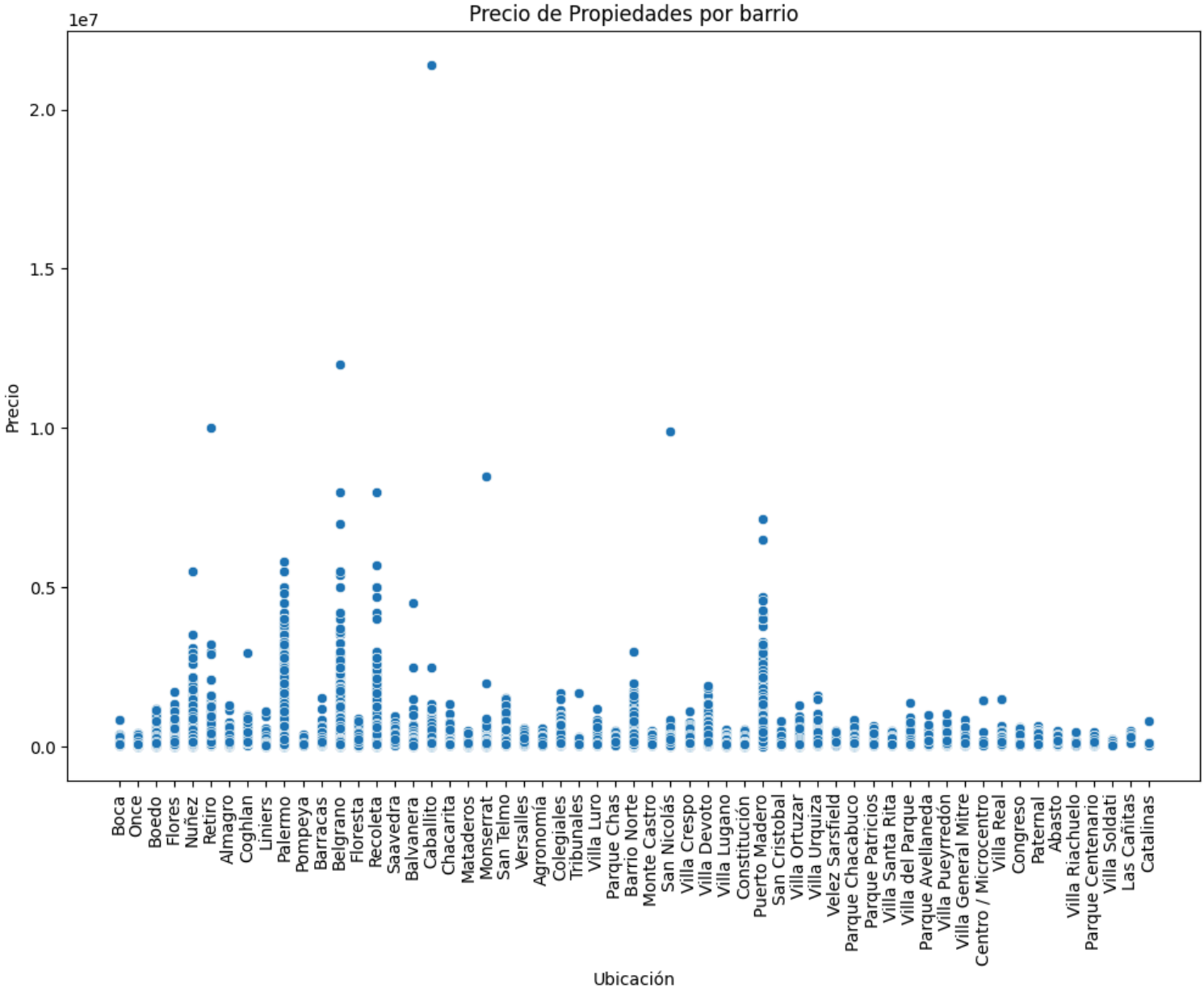
En el caso de las variables multivariadas, un scatterplot nos muestra outliers entre la superficie total de las propiedades y los precios:



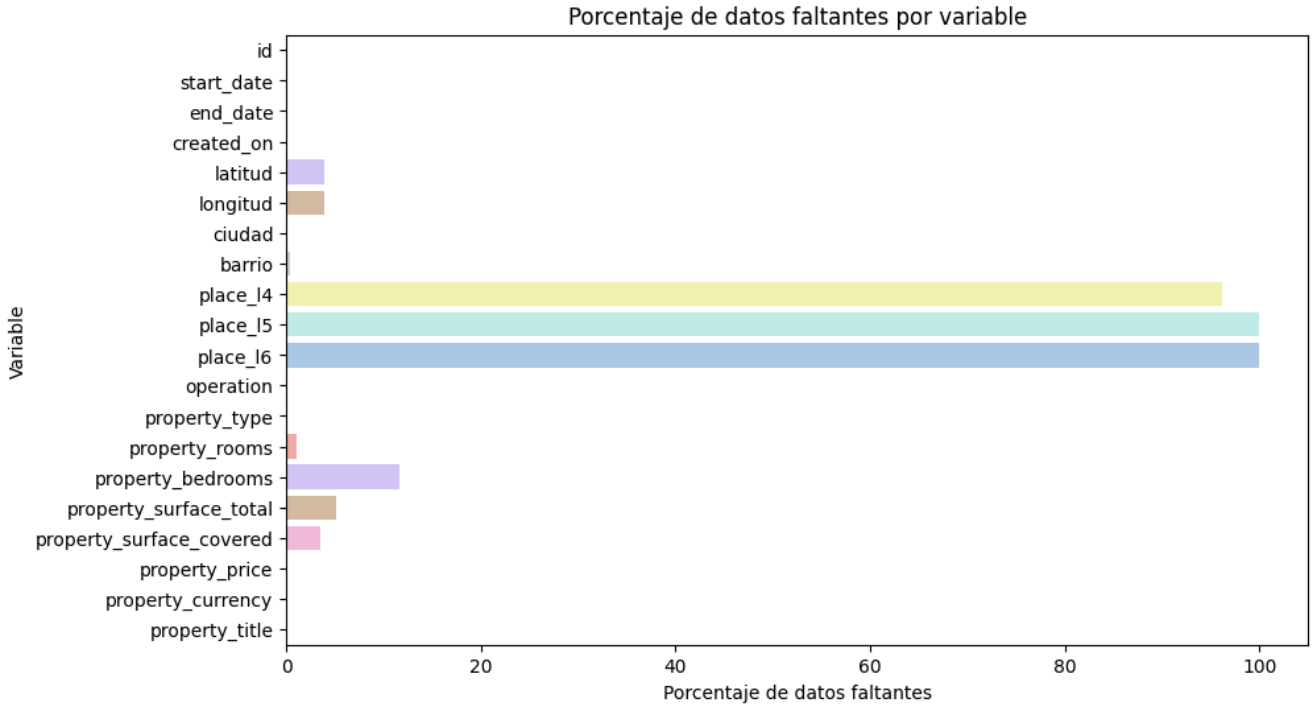
Aquí podemos observar lo dicho en la sección de correlaciones. Los outliers quizá causan una disparidad entre las correlaciones de estos 2 features, a pesar de que la lógica nos indica que están altamente relacionados. Una vez tratados los outliers, el gráfico quedó así:


Relación entre precio de venta y metros de superficie
Precio de venta
20.0M
15.0M
10.0M
5.0M
0
0
100
200
300
400
500
Metros de superficie

Si queremos observar los precios de las propiedades según el barrio en el que se encuentran, podemos observar algunos valores atípicos aquí también:



Con este gráfico podemos observar todas las columnas de interés, y sus respectivos datos faltantes:



Los features cómo “**place_l4**”, “**place_l5**”, “**place_l6**” los consideramos con demasiados datos faltantes como para poder ser tratados y utilizados, de esta forma, decidimos removerlos del dataset y trabajar sin ellos, ya que, en caso de solucionar estos problemas de alguna forma, como por ejemplo, promediando con aquellos datos que poseemos, obtendremos resultados que no consideramos reales o demostrativos del dominio con el que estamos trabajando.

Para el caso de la **‘longitud’** y la **‘latitud’**, decidimos utilizar el dato del barrio en el que se encuentran esas propiedades, y llenar esos datos con la información aproximada de dónde pueden llegar a encontrarse, basándonos en las latitudes y longitudes de otras propiedades que se encuentran en esos mismos barrios.

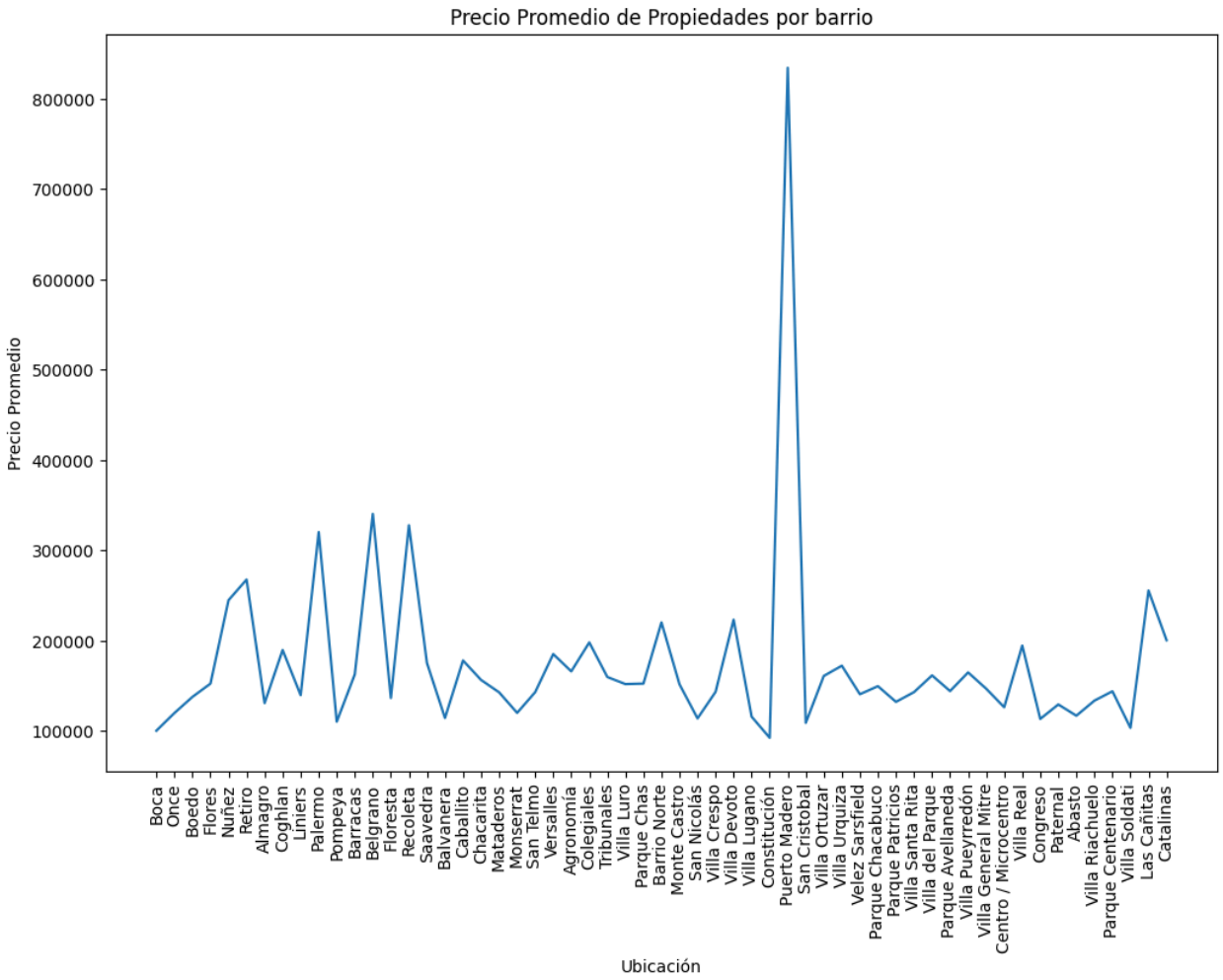
Para las propiedades que tienen la coordenada **‘barrio’** vacía, buscamos propiedades cercanas y les copiamos el barrio.

En cuanto a los valores faltantes en la columna **‘rooms’**, completamos con el promedio de cantidad de habitaciones en las propiedades que sí tienen esta información.

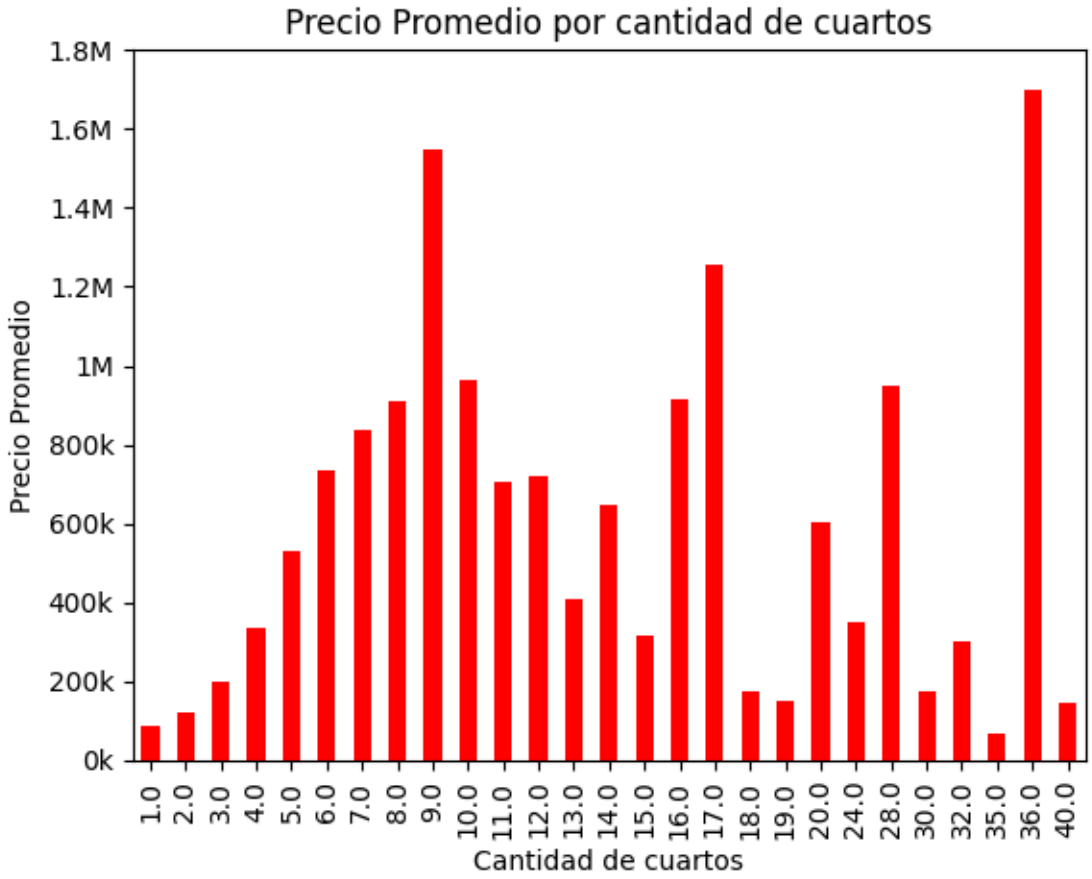
Para la columna de **‘property_bedrooms’**, completamos los valores faltantes con la cantidad de cuartos que tiene esa propiedad - 1. Ya que, en general, en una propiedad se cuentan como ambientes todas las habitaciones más el living.

Por último, completamos los valores faltantes de los atributos **'surface_total'** y **'surface_covered'** usando la técnica de imputación por vecinos más cercanos(KNN). Usamos el valor de vecinos por defecto (n = 5).

Visualizaciones



En este gráfico podemos observar el precio de las distintas propiedades según el barrio en donde se ubican, como era de esperarse las propiedades ubicadas más al norte de CABA son más costosas de las que se encuentran en el sur de la capital.

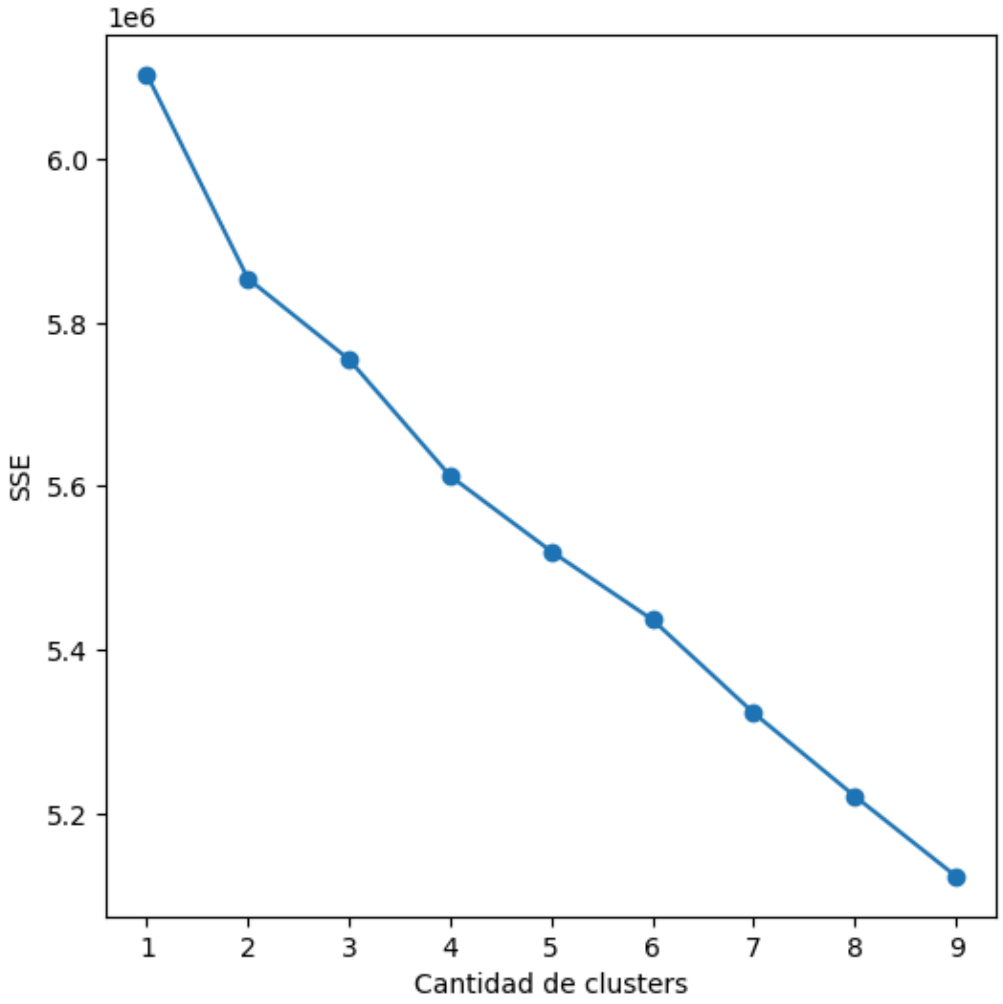


En este otro gráfico observamos la relación precio y cantidad de cuartos. Por un lado, llegamos a la conclusión de que en general cuantos más cuartos tengan más caras serán esas propiedades con algunas excepciones que pueden deberse a la ubicación de dichas propiedades. Por otro lado, también se puede deducir la presencia de algunos datos mal ingresados ya que por ejemplo en nuestro gráfico aparece un promedio de que una propiedad con 40 habitaciones sale menos de 200k USD, en ese caso nos podemos imaginar que a la persona que cargó ese dato se le escapó un 0 y esa propiedad en realidad tiene 4 cuartos.

Clustering

Antes de empezar nuestro proceso de clusterización lo que hicimos fue transformar las columnas con valores no numéricos a valores numéricos para poder utilizarlos. Para hacer esto, usamos el método del 'One Hot Encoding' y transformamos los barrios y los tipos de propiedad en booleanos para poder utilizarlos. Además, eliminamos algunas columnas que no nos son utiles para este analisis: 'start_date', 'end_date', 'id', 'created_on', 'property_title' y filtramos nuestras propiedades en base a las cuales, según su latitud y longitud solo se encuentran dentro de CABA. Como última cosa antes de agrupar nuestro dataset, estandarizamos sus valores con un StandarScaler.

Luego, para decidir cuántos clusters usar, recurrimos al método del codo. Este nos ayuda a decidir viendo en qué número "se quiebra" el gráfico:



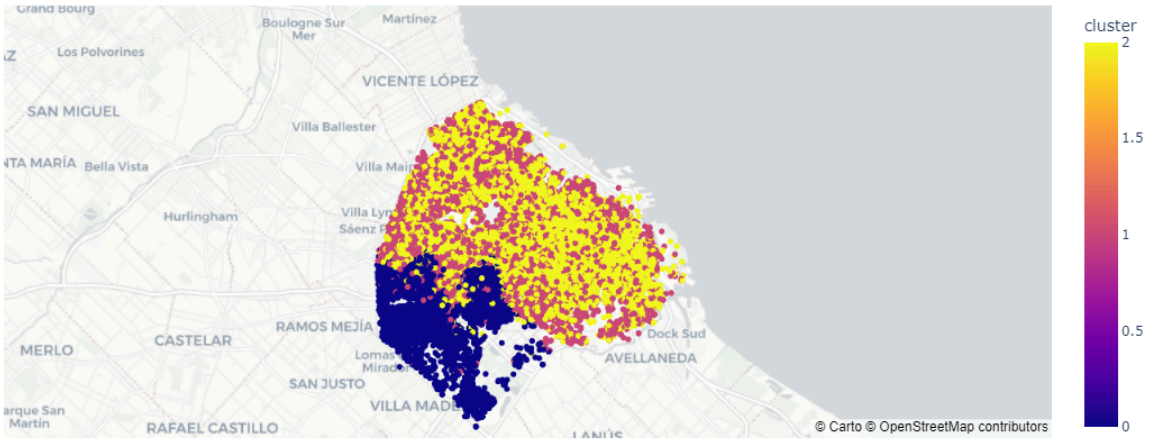
Para tener una “segunda opinión” de cuantos clusters utilizar, también analizamos el número de silhouette para cada cluster. Finalmente, con ambos métodos llegamos a la decisión de usar 2 clusters.

Ahora, con nuestro dataset agrupado en 2 clusters analizamos cuál es el “criterio” que utilizan para agruparse según este mapa de CABA.

Luego de realizar varios gráficos según cantidad de habitaciones, superficie cubierta, en que barrio se ubican cada propiedad y a cluster pertenece llegamos a la conclusión que tenemos una clusterización donde aquellos elementos que pertenecen al grupo azul(0) suelen encontrarse en la zona céntrica, mientras que aquellos que pertenecen al grupo amarillo(1) generalmente se van a encontrar mas cerca de la periferia.

Otra manera de agrupacion que podemos notar es que los elementos del grupo amarillo(1) tienen una mayor superficie total y una mayor cantidad de habitaciones.

Luego, repetimos el análisis anteriormente detallado pero para 3 clusters como pide el enunciado. Veamos como queda distribuido el mapa:



Luego de analizar el mapa y otros gráficos de barras (los mismos vistos para 2 clusters), vemos que el grupo de color azul(0), se concentra en a perfieria de la Ciudad de Buenos Aires, tambien vemos que esta zona “nueva” posee propiedades de precios mas bajos y con una mayor cantidad de casas. Los otros dos grupos parecen respetar la separación que vimos cuando utilizamos únicamente 2 clusters. Vemos una clara tendencia del grupo formado sobre la zona céntrica y cercana al río(amarillo) con un aumento en los precios de dichas propiedades, como es el caso del cluster que tiene la mayor cantidad de datos sobre zonas como 'Recoleta' o 'Puerto Madero'.

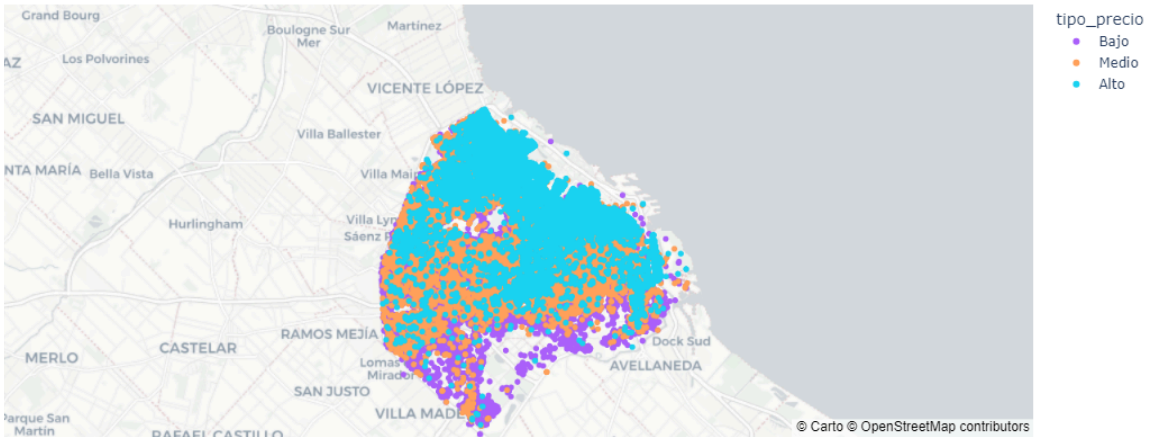
Como conclusión, con 3 grupos vemos una agrupación más clara y distintiva entre las distintas propiedades del dataset.

Clasificación

Mencionar cuál es la alternativa seleccionada para construir la variable “tipo_precio” justificando su elección. Mostrar en un mapa de CABA los avisos coloreados por tipo_precio.

Para construir la variable ‘tipo_precio’, consideramos que es mejor utilizar la división de cuantiles de la forma 25/50/25 sin diferenciar las propiedades por tipo. Llegamos a esta decisión debido a que cuando separamos nuestro análisis por tipo de propiedad, los casos de *departamento* y *PH* tienden a tomar valores de precios altos. Esto puede ser debido a un mal tratado de outliers de nuestra parte, o mismo porque la distribución no es tan efectiva como la elegimos.

A continuacion tenemos un mapa de CABA con las propiedades coloreadas segun su ‘tipo_precio’:



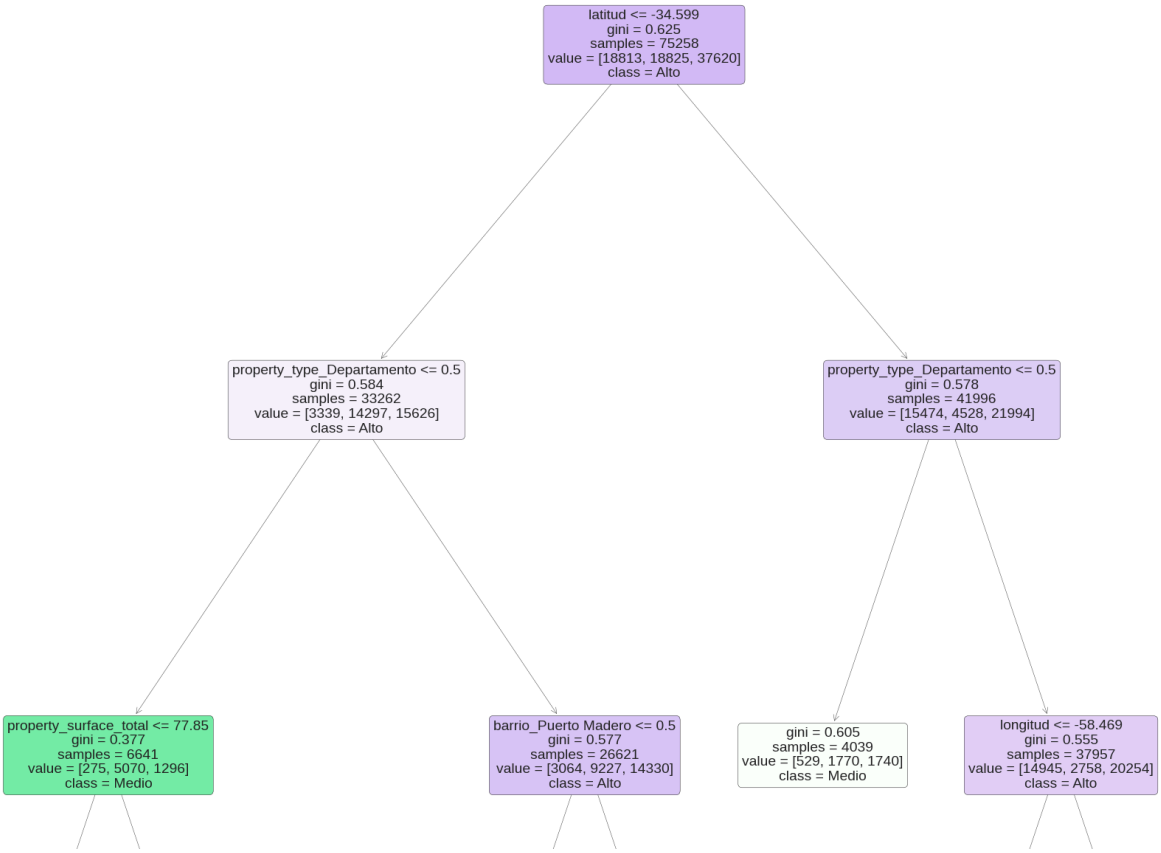
Esta distribución tiene una similitud con el análisis hecho por K-means para 3 grupos ya que, podemos ver que los avisos de precio 'bajo' se encuentran, casi todos, al sur de la ciudad. Mientras que los de precio 'medio' y 'alto' están ubicadas en la zona más céntrica y cercana al río. Además, podemos ver que cuanto más al norte está la propiedad va aumentando el valor de la misma.

a. Construcción del modelo

Para la construcción los modelos, Comenzamos transformando variables categóricas en variables dummy y eliminando columnas irrelevantes como id, start_date, end_date, created_on y property_title. Definimos tipo_precio como la variable objetivo y excluimos variables que podrían inducir sobreajuste, como pxm2 y property_price. Luego, dividimos el conjunto de datos en entrenamiento y prueba utilizando train_test_split.

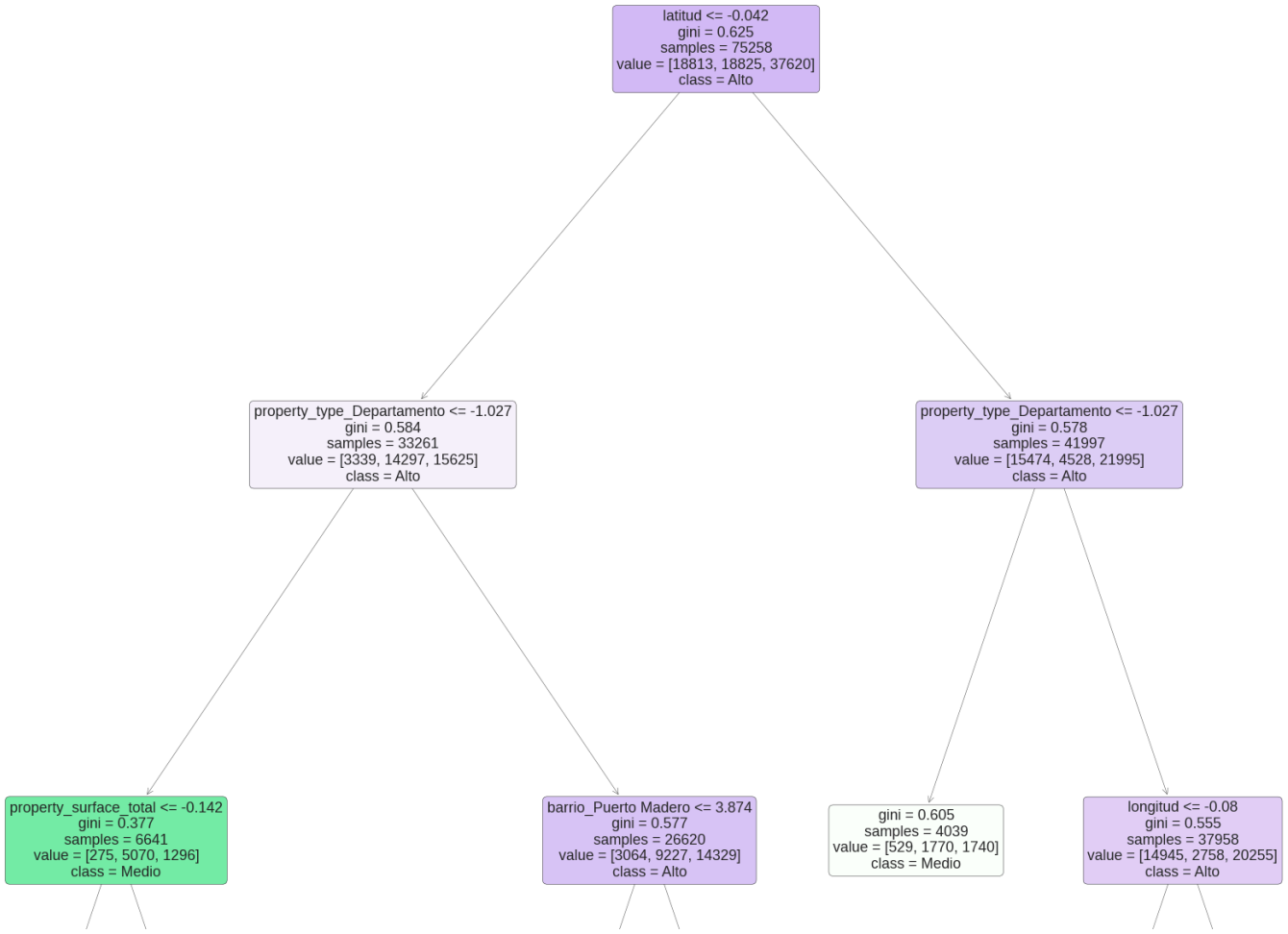
Arbol de Decision

Para evitar el desbalanceo, utilizamos stratified K-fold, donde optamos por utilizar 10 folds. Para la busqueda de hiperparametros, utilizamos la metrica de recall ya que en nuestro contexto, queríamos asegurarnos de que el modelo fuera capaz de identificar correctamente la clase positiva (por ejemplo, propiedades con un cierto tipo de precio), incluso si esto significaba tener algunos falsos positivos adicionales. Por lo tanto, optimizamos los hiperparámetros del modelo para maximizar el recall utilizando RandomizedSearchCV. Los hiperparámetros que buscamos maximizar fueron: 'criterion' con posibles valores 'entropy' y 'gini', 'max_depth' con un rango de 1 a 25 y por último el parámetro 'ccp_alpha' con valores de 0 a 0.05.



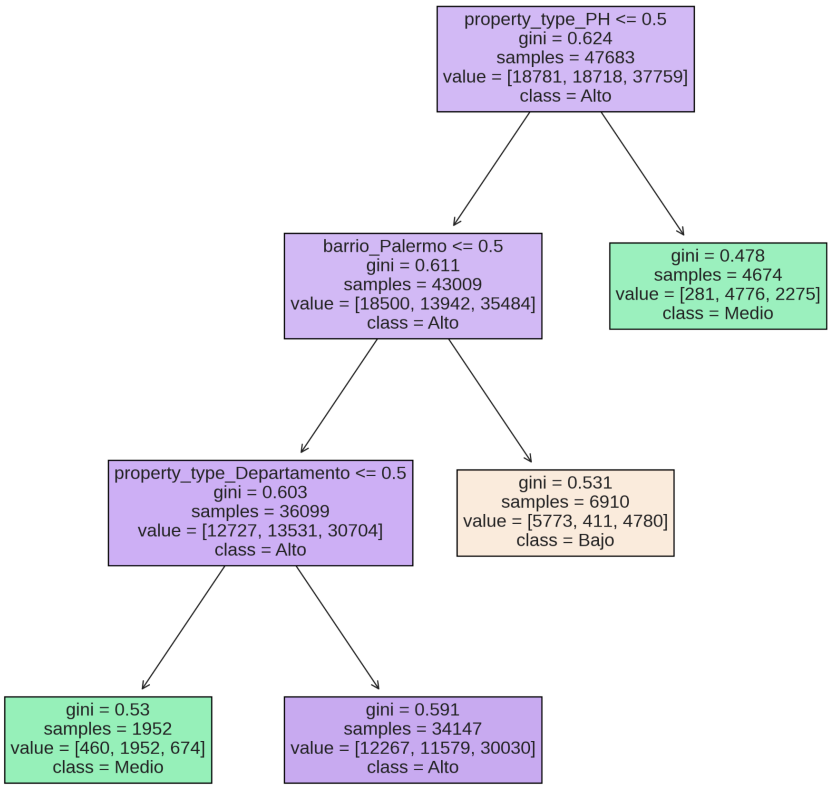
descripción: Observamos una clara importancia sobre la latitud, longitud y los barrios a los que pertenecen las observaciones. Obviamente la latitud, longitud, y los barrios, están directamente relacionados, por lo que podemos concluir que la ubicación de la propiedad es, en gran parte, lo que le otorga su valor. También observamos que la superficie total y si la propiedad es un departamento tienen especial relevancia en este árbol específicamente.

Para finalizar, hicimos un nuevo modelo con los datos normalizados, y no observamos diferencias significativas en las métricas de nuestro modelo. Dejamos aquí el árbol con los datos normalizados.


latitud <= -0.042
gini = 0.625
samples = 75258
value = [18813, 18825, 37620]
class = Alto
property_type_Departamento <= -1.027
gini = 0.584
samples = 33261
value = [3339, 14297, 15625]
class = Alto
property_type_Departamento <= -1.027
gini = 0.578
samples = 41997
value = [15474, 4528, 21995]
class = Alto
property_surface_total <= -0.142
gini = 0.377
samples = 6641
value = [275, 5070, 1296]
class = Medio
barrio_Puerto Madero <= 3.874
gini = 0.577
samples = 26620
value = [3064, 9227, 14329]
class = Alto
gini = 0.605
samples = 4039
value = [529, 1770, 1740]
class = Medio
longitud <= -0.08
gini = 0.555
samples = 37958
value = [14945, 2758, 20255]
class = Alto

Random forest

Para evitar el desbalanceo, utilizamos stratified K-fold, donde optamos por utilizar 5 folds. Para la búsqueda de hiperparámetros, utilizamos la métrica Recall nuevamente. Optimizamos los hiperparámetros del modelo de Random Forest utilizando K-fold Cross Validation. Los hiperparámetros optimizados fueron: criterion (con valores 'gini' y 'entropy'), max_depth (con valores de 1 a 19), ccp_alpha (con valores entre 0 y 0.05) y n_estimators (con valores entre 1 y 74). Esto nos permitió encontrar la combinación óptima que maximiza el rendimiento del modelo.



descripción:

Vemos que, similar a nuestro modelo del árbol de decisión, este árbol del random forest parece importarle la ubicación de nuestra nueva propiedad. A su vez, en este árbol de nuestro Random Forest específicamente, vemos como nodo separación inicial si nuestra propiedad es un PH o no, lo cual difiere con nuestro árbol previamente hecho en nuestro modelo anterior.

KNN clasifier

Comenzamos la construcción del modelo KNN Classifier mediante la optimización de sus hiperparámetros. Utilizamos K-fold Cross Validation para explorar diversas combinaciones de los mismos, incluyendo el número de vecinos más cercanos (n_neighbors), la función de peso (weights) y el algoritmo utilizado para calcular los vecinos más cercanos (algorithm). Además, en K-fold Cross Validation utilizamos 5 folds para evaluar el desempeño del modelo en diferentes separaciones de datos. Durante la búsqueda de hiperparámetros, utilizamos recall como métrica de evaluación, con el objetivo de encontrar una combinación que maximice el rendimiento del modelo en términos de identificar correctamente la clase positiva, incluso si eso genera algunos falsos positivos.

b. Cuadro de Resultados

Modelo	F1-Test	Precision Test	Recall Test	Accuracy Test	
Arbol de Decision	0.55	0.603	0.556	0.59	
Random Forest	0.513	0.656	0.5	0.6	
KNN classifier	0.706	0.713	0.696	0.72	

Nuestros modelos quedaron de la siguiente forma:

Árbol de decisión: {'max_depth': 11, 'criterion': 'gini', 'ccp_alpha': 0.002631578947368421}

Random Forest: {'n_estimators': 68, 'max_depth': 15, 'criterion': 'gini', 'ccp_alpha': 0.002631578947368421}

KNN Clasificación: {'weights': 'distance', 'n_neighbors': 20, 'algorithm': 'ball_tree'}

En cada caso, nuestra performance en el set de evaluación fue un poco peor que en el set de entrenamiento. Sin embargo, observamos que en el KNN de clasificación las métricas en el conjunto de entrenamiento eran altas, niveles de 0.99 o 0.98 para algunas, mientras que las del conjunto de evaluación son las que se encuentran en el cuadro de resultados. Interpretamos esto como un caso de overfitting, pero como las métricas de este modelo siguen siendo mejores que las del Random Forest y las del árbol de decisión, optamos por quedarnos con este modelo de igual forma. Para poder apoyar incluso más esta decisión, deberíamos tomar todavía más datos, y ver como performa nuestro modelo de KNN classifier con estas nuevas observaciones, para saber si realmente puede predecir en niveles de métricas como los provistos en el cuadro de resultados.

c. Elección del modelo

Luego de los resultados obtenidos con los 3 modelos, decidimos quedarnos con el modelo de KNN de clasificación, que es el que mejores métricas nos dio, y el que parece predecir de manera más acertada nuestras observaciones. Reiteramos la necesidad de ver todavía más observaciones nuevas para evaluar y tener más soporte sobre nuestra elección de modelo.

Regresión

a. Construcción del modelo

KNN

Para la construcción del modelo KNN Regresión, implementamos K-fold Cross Validation con 5 folds para evaluar su desempeño y optimizar los hiperparámetros. Utilizamos la métrica de error absoluto medio (mean_absolute_error) para buscar la combinación óptima de hiperparámetros que minimizará los errores en las predicciones del modelo .

Los hiperparametros que intentamos optimizar fueron:

- ‘n_neighbors’ Con valores [1,5,10,15,20,25]
- ‘weigths’ con 2 posibilidades: uniforme y distancia.

Luego de ajustar el modelo, evaluamos su rendimiento en el conjunto de evaluación utilizando varias métricas, incluyendo Mean Absolute Error (MAE), Mean Squared Error (MSE) y Root Mean Squared Error (RMSE). Comparamos estos resultados con la performance del modelo en el conjunto de entrenamiento para evaluar su capacidad de generalización a datos no vistos.

Al evaluar sobre nuestros conjuntos de entrenamiento y evaluación, observamos lo siguiente:

Conjunto de entrenamiento:

```
Mean Absolute Error: 3306.904562550607
Mean Squared Error: 343460061.192695
Root Mean Squared Error: 18532.675500118567
```

Conjunto de evaluación:

```
Mean Absolute Error: 37653.45274400571
Mean Squared Error: 18035040721.185505
Root Mean Squared Error: 134294.60421470963
```

Luego de observar estas métricas, estamos claramente sobre un caso de overfitting. Nuestro conjunto realiza predicciones medias para el conjunto de entrenamiento, pero en el conjunto de evaluación las métricas se disparan, siendo 10 veces más altas que en el conjunto de entrenamiento. Nuestro modelo no pudo encontrar las relaciones generales entre las variables de nuestras observaciones. Intentamos reducir la cantidad de vecinos que toma el modelo, para intentar evitar este overfitting, pero no obtuvimos buenos resultados. Concluimos que quizá se deba a la normalización de nuestros datos o simplemente que KNN no es un buen modelo para resolver este problema de regresión.

XGBoost

Para la construcción del modelo XGBoost, utilizamos la métrica de error absoluto medio (mean_absolute_error) como función de puntuación en el GridSearchCV para buscar la combinación óptima de hiperparámetros que minimizan estos errores.

Los hiperparametros que buscamos optimizar con el GridSearchCV fueron:

- 'learning_rate' : con valores: [0.1, 0.2].

- 'max_depth' : con valores : [5, 10, 15, 20].
- 'n_estimators' : con valores: [5, 10, 15, 20].
- 'lambda': con valores: [0.1, 0.2].
- 'alpha': con valores: [0.01, 0.02].

Luego de ajustar el modelo, evaluamos su rendimiento en el conjunto de evaluación utilizando varias métricas, incluyendo Mean Absolute Error (MAE), Mean Squared Error (MSE) y Root Mean Squared Error (RMSE).

Al evaluar sobre nuestros conjuntos de entrenamiento y evaluación, observamos lo siguiente:

Conjunto de entrenamiento:

```
Mean Absolute Error: 10889.81268989364
Mean Squared Error: 824704685.3446672
Root Mean Squared Error: 28717.672004267115
```

Conjunto de evaluación:

```
Mean Absolute Error: 38628.35695464058
Mean Squared Error: 21270985939.154434
Root Mean Squared Error: 145845.76078568219
```

Concluimos después de observar las métricas, que nuestro conjunto es un mal predictor. Si bien la diferencia entre las métricas de entrenamiento y evaluación ya no son de 10 veces más, sigue habiendo una diferencia, pero ahora sumamos valores medios más altos en nuestros errores. Definitivamente no es un caso de overfitting, pero tampoco tenemos mejores resultados que con el modelo de KNN regression.

Probamos con duplicar la cantidad de estimadores y el máximo de profundidad que podían tener, intentado mejorar nuestros resultados, conseguimos lo siguiente:

Conjunto de entrenamiento:

```
Mean Absolute Error: 6448.640527029726
Mean Squared Error: 711720672.3984885
Root Mean Squared Error: 26678.093492573425
```

Conjunto de evaluación:

```
Mean Absolute Error: 39608.61948993489
Mean Squared Error: 21942466188.937263
Root Mean Squared Error: 148129.89633742833
```

Conseguimos mejores resultados, pero vemos quizá un caso que parece ser overfitting, aumentamos nuestros parámetros de regularización para ver si encontramos un mejor balance. Los resultados obtenidos fueron:

Conjunto de entrenamiento:

```
Mean Absolute Error: 6448.106311930189
Mean Squared Error: 711719253.9099654
Root Mean Squared Error: 26678.066907292316
```

Conjunto de evaluación:

```
Mean Absolute Error: 39611.60241169155
Mean Squared Error: 21942748040.22616
Root Mean Squared Error: 148130.84769968124
```

Conseguimos valores muy parecidos a los del anterior XGboost, por lo que concluimos que nuestro modelo cayó en un caso de overfitting. Sin embargo, parecería ser un poco mejor que nuestro modelo de KNN. Nos quedamos con el primer XGboost que construimos como modelo actual.

Adaboost

Para evaluar el modelo de AdaBoostRegressor, se empleó GridSearchValidation. Durante la búsqueda de hiperparámetros mediante Grid Search, se utilizó la métrica de error absoluto medio (MAE) para seleccionar la combinación óptima de parámetros.

Los hiperparametros que fueron optimizados son:

- 'learning_rate' : con valores: [0.1, 0.2],
- 'n_estimators' : con valores en un rango de 1 a 16.
- 'loss' : con valores: [linear, square],

Posteriormente, se evaluó el rendimiento del modelo en el conjunto de evaluación utilizando varias métricas, como el MAE, el MSE y el RMSE. Estos indicadores proporcionan una comprensión completa del desempeño del modelo.

Al evaluar sobre nuestros conjuntos de entrenamiento y evaluación, observamos lo siguiente:

Conjunto de entrenamiento:

```
Mean Absolute Error: 88825.96893685969
Mean Squared Error: 38532062395.767685
Root Mean Squared Error: 196295.85425007754
```

Conjunto de evaluación:

```
Mean Absolute Error: 88447.65147439433
Mean Squared Error: 42299170364.00589
Root Mean Squared Error: 205667.62108802126
```

Este parece ser el modelo con la menor diferencia de los 3 que hemos entrenado. Sin embargo, el problema de este modelo son los altos valores a la hora de realizar sus predicciones. Para nuestro problema de regresión, que una propiedad tenga más, o menos de 80.000 dólares en nuestra predicción es algo grave, así que no podemos tenerlo en cuenta.

b. Cuadro de Resultados

Modelo	MSE	RMSE	MAE
KNN	343460061.19	18532.67	3306.90
XGBoost	21270985939.15	145845.76	38628.35
Adaboost	42299170364	205667.62	88447.65

Nuestros modelos quedaron de la siguiente forma:

KNN de Regresión: {'weights': 'distance', 'n_neighbors': 25}

XGBoost: {'alpha': 0.02, 'lambda': 0.2, 'learning_rate': 0.2, 'max_depth': 20, 'n_estimators': 20}

KNN Clasificación: {learning_rate= 0.1, loss= 'square', n_estimators= 11}

c. Elección del modelo

Como conclusión final, para nuestros problemas de regresión, decidimos quedarnos con el XGBoost como modelo para predecir. La razón detrás de nuestro razonamiento es que es un caso intermedio entre los otros 2 modelos. Tiene menor error en módulo que nuestro adaboost, y tiene menor distancia entre nuestras métricas de entrenamiento y nuestras métricas de evaluación, por ende nos parece el mejor.

Conclusiones Finales

Luego de realizar el trabajo, podemos llegar a varias conclusiones. Para empezar, el trabajo sobre los datos se realiza en todas las etapas del trabajo, ya sea normalizando, encontrando nuevos outliers que previamente no pudimos ver, o haciendo One Hot Encoding, entre otros ejemplos posibles. Por otro lado, los problemas de clasificación parecen ser un poco menos costosos y requieren menor trabajo de los features. Los problemas de regresión creemos que fallaron por algunas razones, las principales siendo que no tratamos los outliers de una forma más profunda, podríamos haber realizado un mejor trabajo allí, y también nos faltaron features que pudieran otorgar información sobre las observaciones que no armamos. Ya sea distancias a espacios verdes, estado interno de las propiedades, cercanía a zonas urbanas importantes como Retiro o Constitución, entre otros posibles features que no pudimos realizar. Nuestra búsqueda de features fue superficial y por ende nuestras predicciones en la regresión también lo fueron.

Nos hubiera gustado conseguir un modelo de regresión que tuviera mejores métricas, dado que conseguir estas métricas hubiera requerido de nosotros realizar un mejor trabajo con los datos, y quizá hubiéramos descubierto información interesante sobre estos datos que no podríamos haber predicho al iniciar este trabajo.

Tiempo dedicado

Integrante	Tarea	Prom. Hs Semana
Catalina Basso	Filtrado inicial. Análisis de variables cualitativas y cuantitativas. Variables Irrelevantes. Visualización de datos. Análisis, decisión y ejecución sobre datos faltantes a nivel columna y fila. Informe chp1. Análisis de en cuántos clusters agrupar. Análisis de calidad de los clusters. Ordenar colab. Informe chp2. Correcciones sobre el chp1 y chp2. Construcción del modelo KNN. Informe entrega final.	7
Lucas Ruiz	Gráficos de las distribuciones de las variables más importantes (antes y después de ser tratadas). Correlaciones. Visualizaciones. Visualización de datos faltantes a nivel columna y fila. Análisis y visualización de valores atípicos(univariados y multivariados). Ejecución de la decisión sobre cómo tratar los outliers.	9

	Análisis de la relación entre el precio de venta y los metros de superficie. Análisis de cada grupo intentando entender en función de qué características fueron formados. Gráfico de los grupos sobre un mapa de CABA. Colaboración al crear la variable tipo_precio según el análisis del precio por metro cuadrado de las propiedades. Colaboración al comparar la variable tipo_precio y la agrupación por K-means. Construcción, entrenamiento y evaluación del Modelo a elección (AdaBoost) y prueba de distintos parámetros para optimizar el modelo y mejorar los resultados.	
Cristobal Alvarez	Análisis de outliers. Tratado de outliers. Gráficos de variables. Armado de informe. Análisis de tendencia al clustering de nuestro dataset. Estimación de la cantidad apropiada de grupos a utilizar. Evaluación de la calidad de los grupos formados. Gráfico de los grupos sobre un mapa de CABA. Creación de la variable tipo_precio según el análisis del precio por	16

	metro cuadrado de las propiedades. Comparación entre la variable tipo_precio y la agrupación por K-means. Obtención de HiperParametros para la creación de todos los modelos entrenados durante el trabajo. Entrenamiento de todos los modelos, evaluación de los mismos, resultados y conclusiones de todos los modelos. variación en los distintos modelos para buscar mejores resultados. Tratado de los datos para poder trabajar con los modelos. Conclusiones.	
Dalmiro Vilaplana	Análisis de correlaciones. Visualización de datos. Gráfico de distribución de las variables importantes. Análisis de outliers. Análisis de clusters. Correcciones chp 1. Ejecución del Z-score para el nuevo análisis de outliers obtención de hiperparametros armado del reporte.	5