Checkpoint 2 - Grupo 06

Análisis Exploratorio

Descripción del dataset

El dataset provisto consiste de una lista de anuncios de propiedades en venta o en alquiler, generado en el año 2021 por la empresa Properati. Las propiedades se encuentran mayormente en Argentina, pero es posible identificar algunas de ellas ubicadas en Uruguay, Brasil, e incluso Estados Unidos. Se incluyen desde avisos de casas y departamentos, hasta locales comerciales y casas de campo.

El dataset completo se compone de 460.154 registros, con información distribuida en 20 columnas, las cuales pueden distinguirse entre cualitativas o cuantitativas según la naturaleza del dato y su rol en el modelo.

Variables cualitativas:

- ***id*:** cadena de caracteres única para identificar cada publicación.
- ***start_date**:* fecha en la que el anuncio se abre para la venta/alquiler. Utiliza todas las fechas posibles del año 2021. Coincide con la columna *created_on*.
- ***end_date**:* fecha en la que el anuncio se cierra. Las publicaciones cerradas tienen fechas de 2021 y de 2022. Aquellas que todavía están abiertas tienen el valor representativo '9999-12-31'.
- ***created_on**:* fecha en la que se creó la publicación. Coincide con la columna *start_date*.
- ***place_l2*:** detalle de la provincia/departamento donde está la propiedad.
- ***place_l3*:** detalle del partido/barrio/pueblo donde está la propiedad.
- ***place_l4*:** detalle de localidades en un partido.
- ***place_l5*:** detalle para barrios cerrados.
- ***place_l6*:** subdivisión adicional. No es usada por ninguna propiedad.
- ***operation*:** tipo de contrato del aviso. Dependiendo del aviso puede ser venta, alquiler, o alquiler temporal.
- ***property_type*:** tipo de inmueble del aviso. Los valores posibles son departamento, casa, lote, PH, local comercial, oficina, cochera, depósito, casa de campo, u otro tipo sin especificar.

* Las columnas que involucran fechas presentan la ambigüedad de poder tratarse como cualitativas o cuantitativas. Optamos por mantenerlas como cualitativas, dado que el análisis que realizamos no considera las fechas como un factor determinante del precio de una propiedad.

- ***property_currency*:** moneda del precio de la propiedad. Entre las opciones están el dólar estadounidense, el peso argentino, el peso uruguayo, el peso peruano, y el peso colombiano.
- ***property_title*:** título del aviso cargado en la página web. Es personalizable por aviso, aunque muchos utilizan un título por defecto con información mínima sobre la propiedad.

Variables cuantitativas:

- ***latitud*:** coordenada de latitud de la propiedad en cuestión.
- ***longitud*:** coordenada de longitud de la propiedad en cuestión.
- ***property_rooms*:** cantidad de ambientes detallados en el aviso.
- ***property_bedrooms*:** cantidad de dormitorios detallados en el aviso.
- ***property_surface_total*:** superficie total que ocupa el terreno donde se encuentra la propiedad.
- ***property_surface_covered*:** superficie cubierta correspondiente a la propiedad.
- ***property_price*:** precio de la propiedad detallado en el aviso.

La consigna del trabajo práctico pide trabajar con una muestra del dataset original, que deberá incluir únicamente ofertas de venta en dólares de PH, casas, o departamentos, ubicados en Capital Federal.

Con estos parámetros, la cantidad de registros del dataset se reduce a **94.249**.

Por último, dividimos el dataset en conjuntos de entrenamiento y de prueba, de forma que se pueda desarrollar el modelo con uno de ellos, y luego evaluar el rendimiento del modelo generalizado a un conjunto de datos independiente.

El criterio para la división fue de 80% (**75.399**) de los registros para el conjunto de entrenamiento, y el 20% (18.850) restante para el conjunto de prueba.

Preprocesamiento de Datos

Selección de variables

Un análisis inicial del contenido del dataset revela que algunas de las columnas no son relevantes para el modelo de predicción.

- La columna *id* contiene una cadena única para cada registro del dataset. Este valor no muestra ninguna relación aparente con el resto de los datos del registro, por lo que puede omitirse del análisis.
- Las columnas utilizadas para el filtro inicial de la información que contenían un solo dato posible ya sirvieron su propósito y pueden ser retiradas. Estas columnas serían *operation* ('Venta'), *property_currency* ('USD'), y *place_l2* ('Capital Federal').
- Los atributos *start_date*, *end_date*, y *created_on*, representan fechas relevantes de cada aviso en el dataset. Sin embargo, para la predicción del precio de venta de una propiedad, estas variables no resultan significativas.
- Las columnas *place_l5* y *place_l6* permanecen vacías en todos los registros del dataset filtrado, por lo que pueden ser retiradas sin complicaciones.

Si bien es posible que el precio de una propiedad se modifique en relación a una fecha determinada, el dataset no contiene información sobre variaciones de precio, y suponemos que el precio que se incluye es aquel del momento en el cual se generó el propio dataset. Es por eso, y para evitar un posible ***overfitting*** en el modelo al incluir patrones temporales, que decidimos retirar las fechas de la predicción, enfocándonos en su lugar en las características físicas y ubicación como principales impulsos detrás del precio de la propiedad.

Luego, las columnas relevantes para el modelo de aquí en adelante son:

latitud, longitud, place_l3 (zone), place_l4, property_type, property_rooms, property_bedrooms, property_surface_total, property_surface_covered, property_price, y *property_title*.

Correlaciones entre variables

En cuanto a correlaciones existentes, se planteó un heatmap entre las variables numéricas del problema, respecto al precio de la propiedad.

- La más evidente ocurre con *property_rooms* y *property_bedrooms*. La correlación es fuertemente positiva (0,87) en estos casos, por lo que el precio de una propiedad es propenso a aumentar en base a estos dos atributos.

Para profundizar en este aspecto, se generaron algunos features, particularmente ***property_bedrooms_ratio*** y ***property_surface_ratio***, para definir si alguna de estas proporciones influye de alguna manera en el precio de la propiedad.

- El coeficiente de correlación entre habitaciones versus ambientes y el precio resultó prácticamente nulo (0,062126).
- De modo similar, el coeficiente entre superficie cubierta versus superficie total y el precio no permitió sacar mayores conclusiones (-0,001285).

Sin embargo, consideramos que es una evaluación temprana y que tras una posterior limpieza de los datos, se podrían observar correlaciones más significativas.

Datos faltantes

Para tratar los datos faltantes, inicialmente se calcularon los porcentajes de datos nulos por columna. La mayoría de estos muy probablemente se deba a faltas al momento de la carga de datos, ya que no todas las variables son requeridas a la hora de publicar un aviso.

- Las variables mencionadas anteriormente que fueron usadas como filtro, no tienen datos faltantes. A estas se suman *id, property_price,* y *property_title*.
- Variables como *latitud, longitud, zone, property_rooms, property_surface_total,* y *property_surface_covered*, tienen alrededor del 5% o menos de datos faltantes.
- Las dos variables restantes, *place_l4* y *property_bedrooms*, tienen porcentajes de datos faltantes más significativas (96,19% y 11,67% respectivamente), por lo que continuaremos con el análisis de estas dos variables en particular.

Los faltantes en ***place_l4*** se explican ya que esta columna no es más que una subdivisión zonal del barrio de Palermo. Para no mantener esta columna, reemplazamos cada instancia de Palermo en la columna *zone*, por su correspondiente subdivisión.

Para los faltantes de ***latitud***, ***longitud*** y ***zone***, se plantearon múltiples escenarios:

- Si el registro contiene un dato de *zone*, pero no *latitud* y/o *longitud*, se reemplazan los datos de latitud y longitud por la media de todas las filas de la misma zona.
- Si el registro contiene *latitud* y *longitud*, pero no *zone*, se calcula la diferencia entre la ubicación del registro y la ubicación media de cada zona. Y se asignará la zona que presente una menor diferencia respecto de la ubicación.
- Si el registro no contiene *latitud*, *longitud*, ni *zone*, ese dato tendrá que quitarse, ya que no pueden hacerse esas asignaciones basándose únicamente en precios o superficies.

Luego de este filtrado, el número de registros pasa a ser **75.267**.

Para los faltantes en ***property_surface_total***, se le asignará teniendo en cuenta el precio medio por metro cuadrado por zona, dado que el dato de *property_price* no tiene faltantes. Mediante una regla de tres simple, se puede estimar la superficie total.

Para los faltantes en ***property_surface_covered***, se supone que serán una fracción del dato de *property_surface_total*. Entonces se calculó el dato de porcentaje de superficie cubierta en una propiedad, por zona, y se usó para imputar la superficie cubierta en cada fila.

Luego, completamos los datos faltantes en ***property_rooms***, según la información de superficie y habitaciones de otras propiedades de la misma zona. La cantidad de habitaciones en cada propiedad puede estimarse como la superficie cubierta de la propiedad, dividido la superficie promedio por habitación en la zona.

De manera similar, ***property_bedrooms*** tiene que ser una fracción de *property_rooms*. Se calculó el porcentaje de dormitorios dadas las habitaciones, de las propiedades de una misma zona. Se usó este porcentaje para estimar la cantidad de dormitorios por propiedad.

Valores atípicos

A la hora de analizar los valores atípicos, se planteó primero un caso trivial.

- Registros con valores numéricos menores a 0. En un caso particular, la cantidad de habitaciones de una propiedad era negativa. Sin embargo, se consideró un error de tipeo ya que el título del aviso contemplaba este valor como positivo.

La identificación de otros valores atípicos se llevó a cabo gráficamente y mediante sucesivos análisis univariados y multivariados.

Para detectar outliers en *property_rooms* y *property_bedrooms*, se realizaron acercamientos con box plots. A simple vista se podían observar registros con mucho más de 3 IQR de distancia del resto de los datos, para dimensionar la presencia de outliers.

Luego se calcularon los correspondientes Z-score y Z-score modificado de *property_rooms* y *property_bedrooms,* para cada uno de los registros. Se consideraron como outliers aquellos registros que cumplían a la vez con las "**reglas de oro**" de ambas métricas (**Z > 3, Zmod > 3,5**).

- Del lado de *property_bedrooms*, el resultado del análisis arrojó 47 registros para descartar, cuyos valores en esta variable oscilaban entre 75 y 6.028 dormitorios. Luego de eliminarlos, si bien quedaban alrededor de 200 registros con un Z-score modificado mayor a 3,5, la descripción de estas publicaciones mayormente avalaba estos valores elevados.
- Por otro lado, el análisis gráfico de *property_rooms* también mostraba una cantidad excesiva de habitaciones en algunos registros. En base a los Z-score, 52 registros fueron descartados.

Como análisis multivariado, aplicamos el método de distancia de Mahalanobis para las variables *property_bedrooms* y *property_rooms*.

- Un scatter plot inicial revelaba la presencia de outliers evidentes, aunque no demasiadas. Un ejemplo particular es una propiedad con más de 70 dormitorios, pero pocas habitaciones.

Tras el cálculo de la distancia para cada registro, un boxplot de esta distancia revelaría una apariencia bastante similar al análisis univariado previo, con algunos outliers extremadamente alejados del rango intercuartílico.

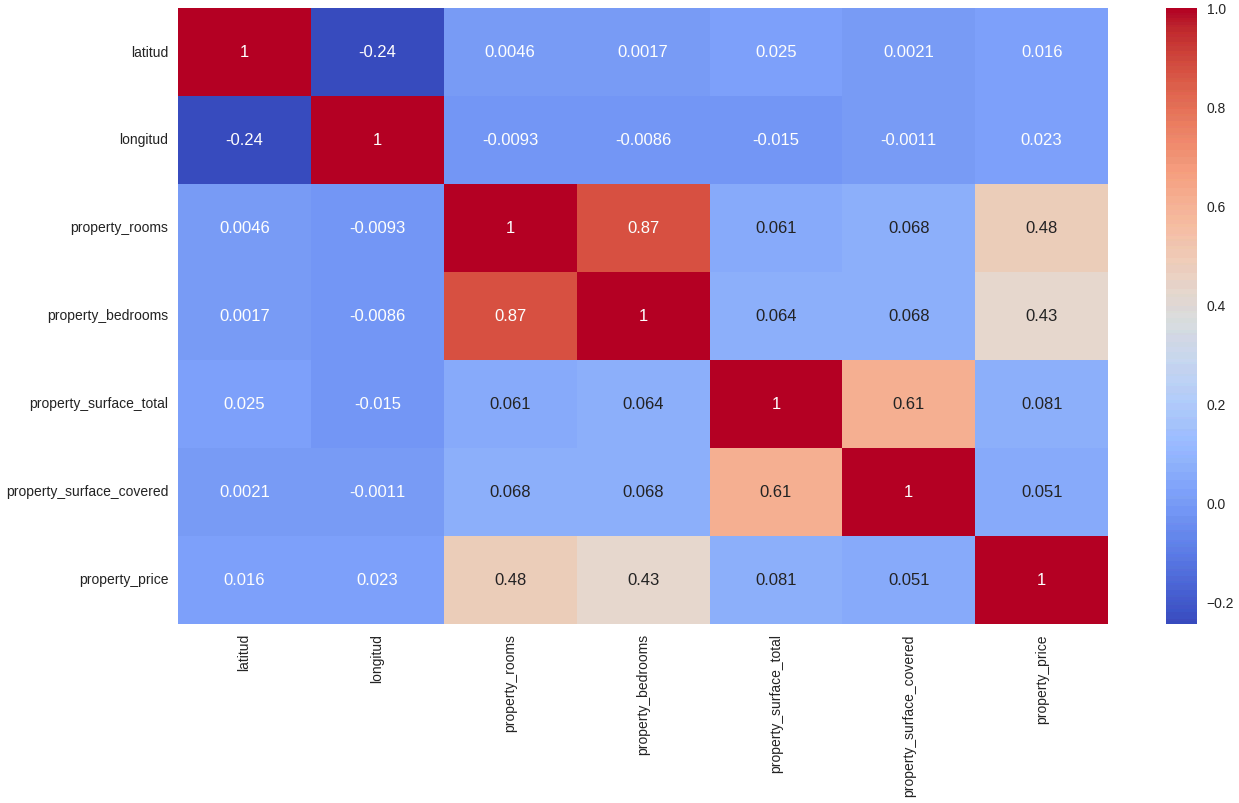
Definiendo un valor umbral de 1100, se aislaron los valores más alejados de la nube de puntos, incluyendo uno que presentaba un error de tipeo. Bajando el umbral a 500, se agregaron algunos valores más cercanos a la nube, pero era razonable considerarlos atípicos. Finalmente decidimos mantener este umbral

más reducido y apartar estos valores del dataset. El scatter plot resultante se adjunta en la sección de visualizaciones.

Posteriormente, al analizar del mismo modo la relación entre las variables ***property_price*** y ***property_surface_total***, seguían existiendo valores anómalos sin detectar. Planteando un umbral de 1.000, se pusieron bajo observación casos bordes, propiedades de muy extensa superficie a precios anormalmente reducidos, y viceversa. Dado que no fue posible definir con seguridad si se trataban de errores de tipeo, o valores reales de algunas zonas particulares, conservamos estos valores en el dataset.

Visualizaciones

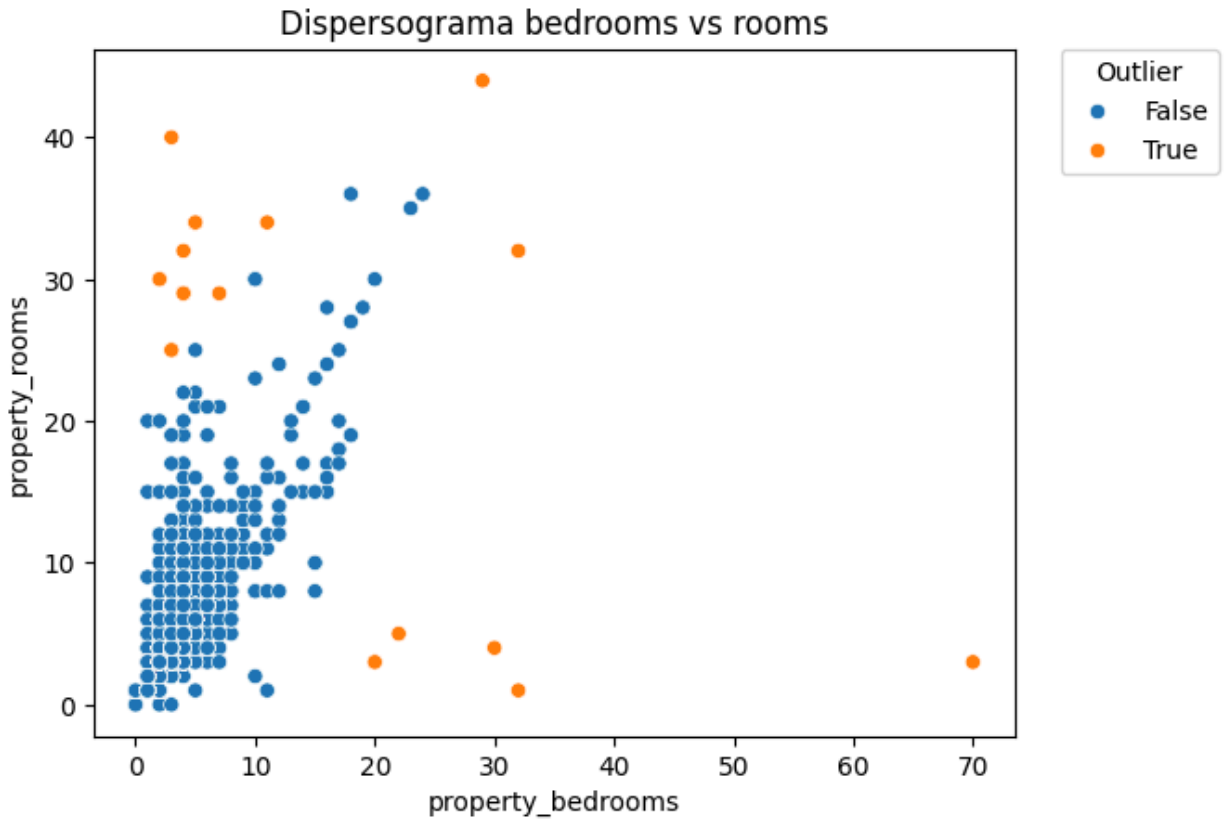
Heatmap de las variables numéricas del dataset, respecto del precio



Decidimos visualizar con un heatmap las distintas correlaciones entre pares de variables cuantitativas del dataset, respecto del precio de la propiedad. Esto es dado que, al tratarse de variables numéricas, las correlaciones pueden identificarse rápidamente. Principalmente se puede observar como *property_rooms* y *property_bedrooms* tienen correlaciones positivas, lo que significa que el precio de la propiedad tiene tendencia a aumentar junto con la cantidad de habitaciones y dormitorios.

La segunda correlación más significativa según el gráfico es la que vincula *property_surface_covered* y *property_surface_total*, indicando que el precio de la propiedad suele aumentar junto a la superficie de ésta, aunque resulta menos influyente que la cantidad de habitaciones.

Scatter plot entre property_rooms y property_bedrooms, distinguiendo outliers según Mahalanobis



El gráfico surgió durante la detección de valores atípicos de forma multivariada. Cada punto representa los valores que cada observación del dataset posee en las columnas *property_rooms* y *property_bedrooms*. La correlación entre estas variables es positiva, teniendo en cuenta que la mayoría del conjunto de datos aumenta las cantidades de forma pareja, con la cantidad de habitaciones superando la de dormitorios consistentemente.

El análisis realizado con las distancias de Mahalanobis permitió distinguir cuáles de estos valores tenían una mayor probabilidad de ser outliers. El valor de umbral utilizado para este gráfico es de 500.

Clustering

Tendencia al clustering

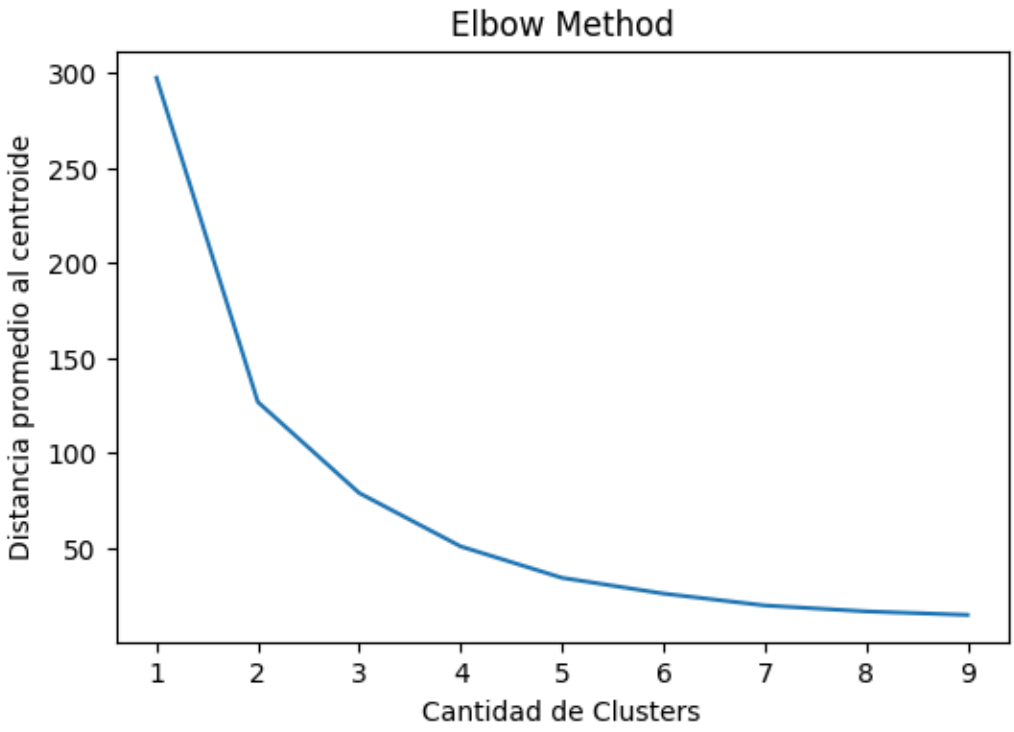
Para el análisis de clustering, se intentó utilizar solamente las columnas *property_rooms* y *property_bedrooms,* ya que son las variables que se probó que exhiben una relación más fuerte con el precio final de la propiedad.

Las demás variables, *latitud, longitud, property_surface_total, property_surface_covered, property_type, y property_price*, estarán a mano para poder tratar de identificar si el algoritmo *K-means* tiene en consideración alguna de estas a la hora de conformar los grupos.

La tendencia al clustering fue calculada mediante la estadística de Hopkins, cuyo resultado fue de un valor próximo a 0.0004. Teniendo en cuenta la hipótesis nula, se puede deducir que no hay clusters significativos en el dataset, y que los datos están relativamente distribuidos de forma uniforme.

Cantidad de clusters

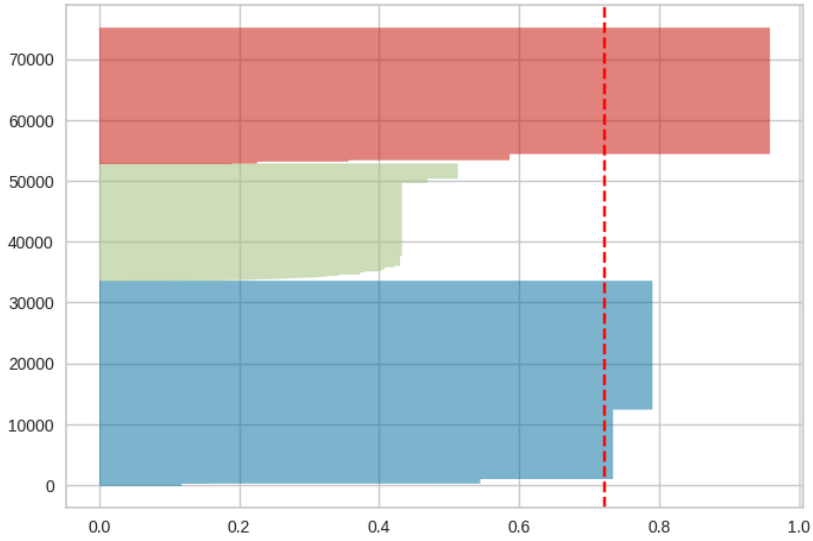
Para tener una idea de la cantidad de grupos a utilizar para la clasificación, primero se planteó el *Elbow method* de forma preliminar. Según su gráfico, obtenido mediante el algoritmo *K-means*, la cantidad recomendada sería de 3 o 4 clusters.

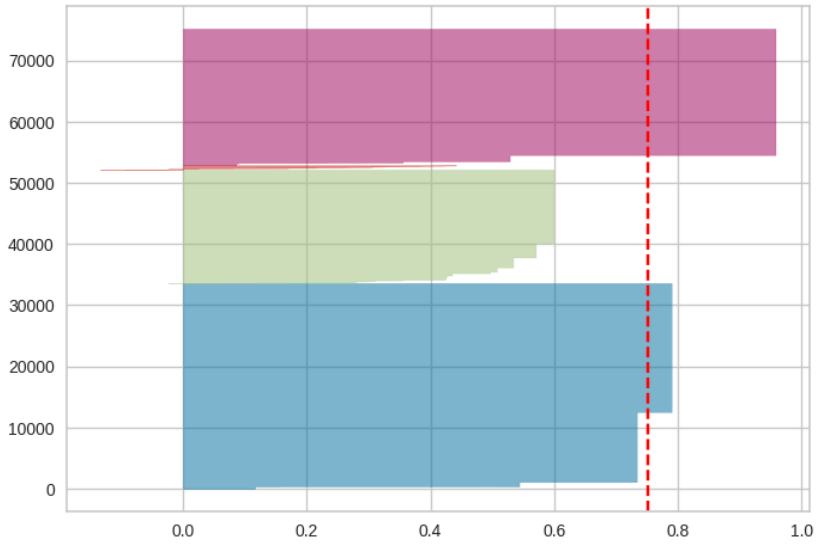


Sin embargo, al analizar la calidad de los grupos con el *Silhouette method*, en un rango de 3 a 6 clusters, el coeficiente favorece los 4 clusters.

- **n = 3**, s(i) = 0.6849078154475335
- **n = 4**, s(i) = 0.7551161921370355
- **n = 5**, s(i) = 0.788372265999519
- **n = 6**, s(i) = 0.7954515274636975

Para definir esta cuestión, generamos los respectivos *silhouette plot*:

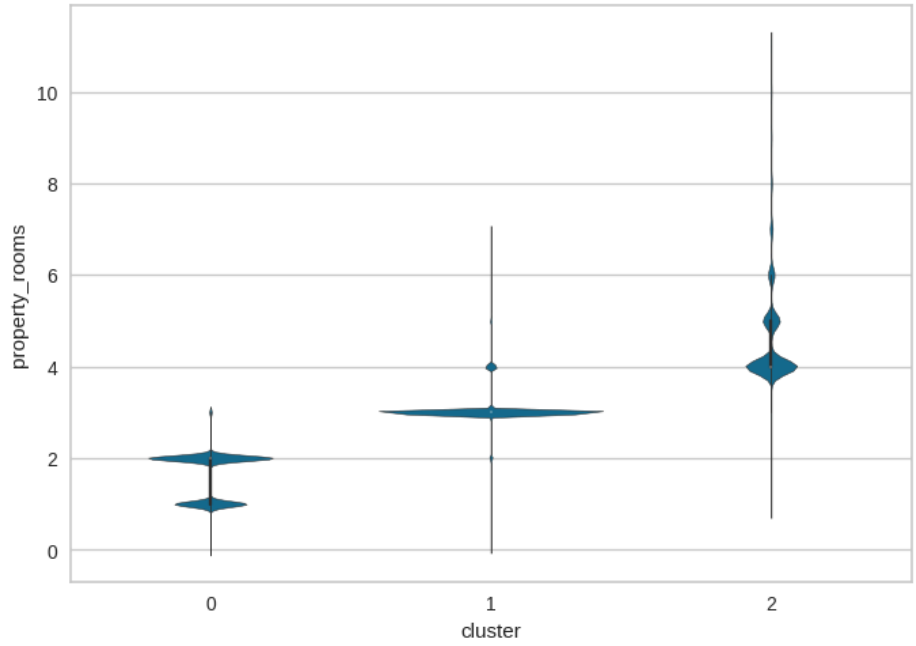


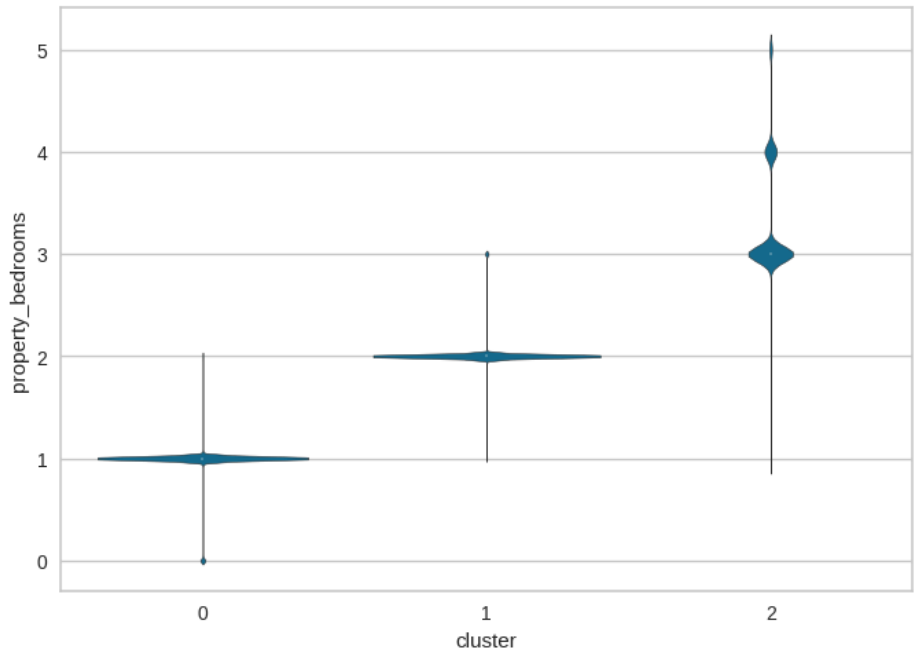


Si bien el gráfico de 4 clusters demuestra coeficientes consistentemente elevados, uno de ellos resulta contener datos con coeficientes bajos y hasta negativos. Esto significaría entrenar un modelo con grupos cuyos datos son potencialmente atípicos, afectando el desempeño de las clasificaciones.

Elegimos entonces la agrupación en 3 clusters tanto por el tamaño más parejo de cada cluster como por la ausencia de clusters de bajo coeficiente.

Realizamos *violin plots* para observar la densidad de los registros según número de habitaciones, y de dormitorios, en cada cluster. De esta forma se buscó comprender cómo fue que se agruparon los datos.



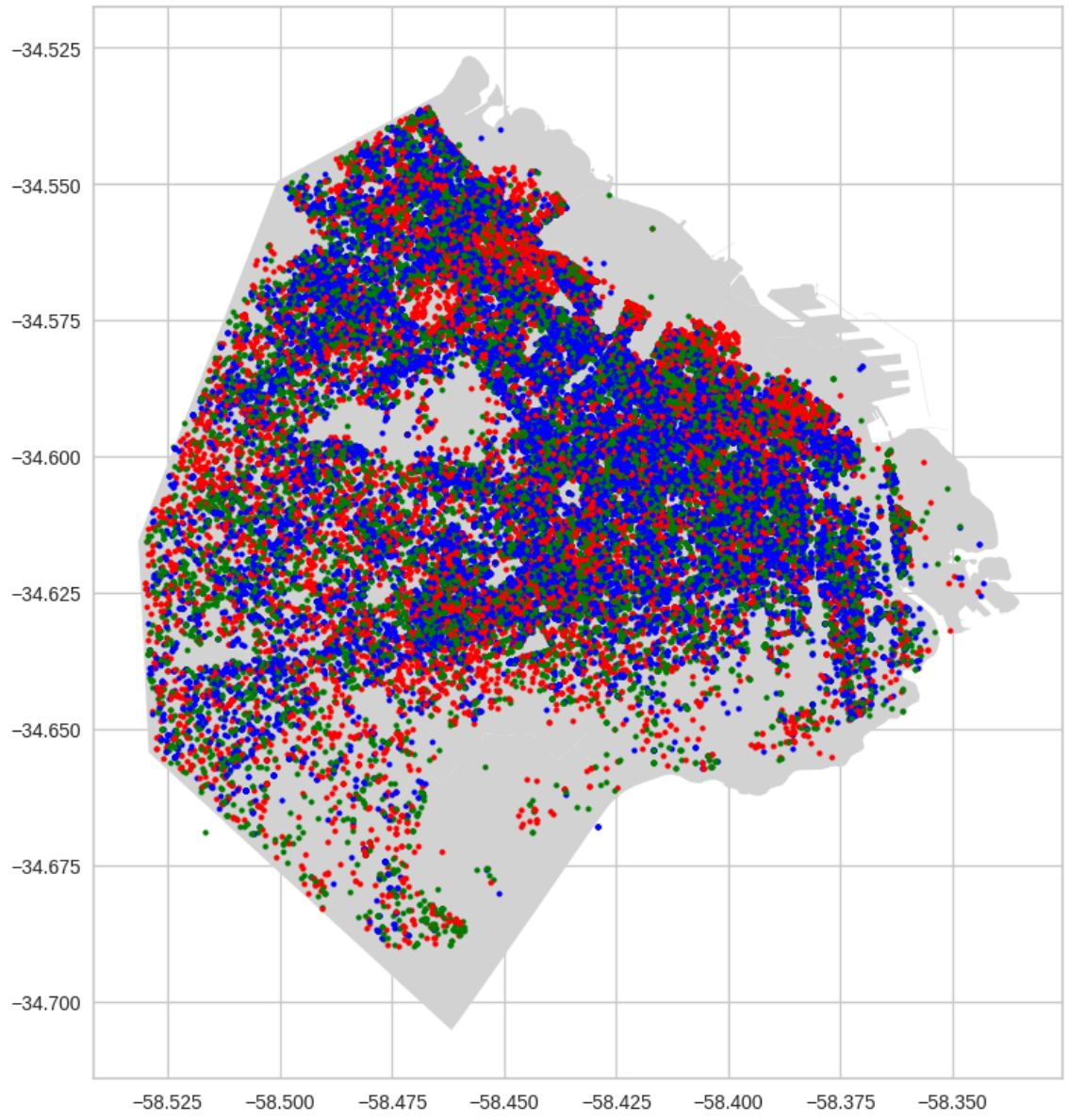


Los clusters están agrupados según la cantidad de habitaciones y dormitorios.

Los grupos 0 y 1 cuentan con una cantidad similar de registros, con la diferencia de que los avisos del grupo 0 detallan menos habitaciones que los del grupo 1.

Luego, el grupo 2 tiene un volumen considerablemente mayor de registros, agrupando aquellos avisos donde se detallan la mayor cantidad de habitaciones.

Llevamos estos grupos a un mapa de CABA para visualizar los avisos del dataset según sus coordenadas de latitud y longitud.



En gran parte de la ciudad, los grupos generados no pueden ser fácilmente identificados según el barrio, están distribuidos sin que la ubicación tenga mucha importancia. Sin embargo, una excepción a esto se presenta en la zona Sur, donde hay una clara falta de propiedades publicitadas, o bien, se encuentran más dispersas que en otras zonas.

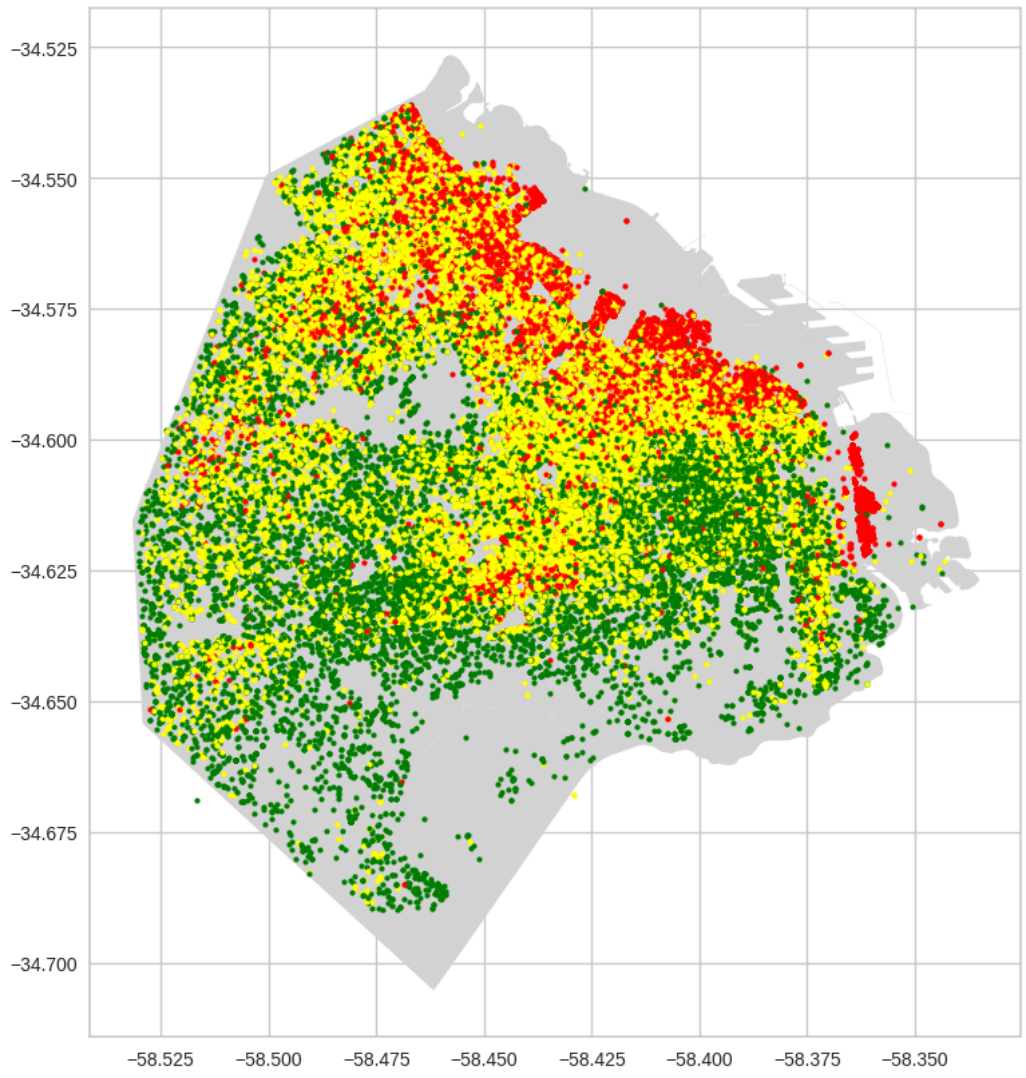
Clasificación

Construcción del target

Entre las alternativas para la creación de la variable ***tipo_precio***, tras calcular la variable precio por metro cuadrado (*pxm2*) de todas las propiedades, optamos por la segunda opción: dividir *pxm2* en 3 intervalos (25% - 50% - 25%) que representen precios bajos, medios y altos.

- La razón de esto fue que, al probar la primera alternativa, tres intervalos iguales, hallamos grandes variaciones entre los precios dentro de cada intervalo, lo que daría resultados no tan óptimos debido al volumen de datos.
- Además, respecto de la tercera alternativa, analizar *pxm2* por tipo de propiedad tampoco resultaba conveniente, ya que al igual que en el caso anterior, las variaciones de precio eran muy diferentes según la propiedad en cuestión, con variaciones muy grandes en los PH, pero más uniformes en el caso de los departamentos.
- Al reducir el tamaño de los intervalos de precios muy altos y muy bajos, las variaciones de estos intervalos quedan más controladas, lo cual nos sirve para definir los límites entre grupos.

Visualizamos estos nuevos grupos en el mismo mapa de CABA que el utilizado durante el problema de clustering.



El gráfico presenta particularidades en cuanto a la distribución de los grupos, según el precio de las propiedades, y se corresponden en cierta medida con los barrios.

a. En principio, el grupo rojo representa los precios altos. Se concentran fuertemente en las zonas Norte (barrios de Palermo, Recoleta y Núñez), y Noreste (barrios de Puerto Madero y Retiro).

b. El grupo amarillo de precios medios está disperso en las zonas más céntricas de la ciudad y alrededores, concentrándose más en la zona Norte.

c. Luego, el grupo verde de precios bajos está más concentrado en las zonas céntricas y hacia el Sur, pero puede encontrarse a lo largo de toda la superficie de la capital.

Si tuviéramos que comparar este gráfico con el realizado con el algoritmo K-means durante la clusterización, podemos observar algunas cuestiones en común.

- La cantidad de grupos coincide, K-means consideró que la cantidad óptima de clusters para este dataset eran 3, según los coeficientes de Silhouette.
- Dado que el algoritmo solo utilizó la proximidad de los datos a los centroides elegidos para construir los clusters, es difícil de determinar cómo se utilizaron los parámetros para agrupar los datos. El análisis de los violin plots indicó que se basó mayormente en la cantidad de habitaciones y dormitorios. La variable tipo_precio, en cambio, explícitamente se basa en el cociente entre el precio listado de la propiedad y la superficie total de esta.
- La distribución por latitud y longitud también presenta diferencias. En el gráfico de tipo_precio, la clase media (amarillo) ocupa una porción considerable del mapa, debido a la estrategia de creación de target empleada, y se observa una distinción por franjas. En el mapa por clusters, sin embargo, los puntos están mucho más distribuidos, demostrando también la variedad de oferta en cuanto a cantidad de ambientes en los avisos.

Construcción del modelo

Para trabajar con los modelos en esta sección tuvimos que realizar ingeniería de características a los dataset del trabajo práctico. Ambos debían tener las mismas features para poder utilizar los modelos. Hicimos diferentes pruebas, pero llegamos a la conclusión de no utilizar el sector de “zona” porque, a pesar de que intentamos transformarla ya sea extrayendo otro feature, por ejemplo “comunas”. o realizando one hot encoding, no observábamos una mejoría en el score sino todo lo contrario. Con la primera solución propuesta perdíamos información al englobar varios barrios en la misma comuna y con la segunda agregamos más columnas que no ayudaban a un mejor desempeño del modelo.

Para la columna “property_title” decidimos obtener una feature numérica, como el tamaño del título del anuncio y la cantidad de palabras, y eliminar la original. Por último para las variables numéricas decidimos escalarlas .

Clasificación por árbol de decisión

Optimizamos los hiperparámetros con la intención de evitar el sobreajuste:

- ➔ **criterion**: el criterio de decisión.
- ➔ **min_samples_leaf**: el mínimo de muestras necesarias para que sea una hoja.
- ➔ **min_samples_split**: el mínimo de muestras para dividir un nodo, es decir para ser considerado una ramificación.
- ➔ **ccp_alpha**: utilizado para la poda del árbol. Valores más altos aumentan la cantidad de nodos podados.
- ➔ **max_depth**: la máxima profundidad del árbol ya que si tenemos un árbol demasiado grande se aprenderá de forma específica los datos del entrenamiento.

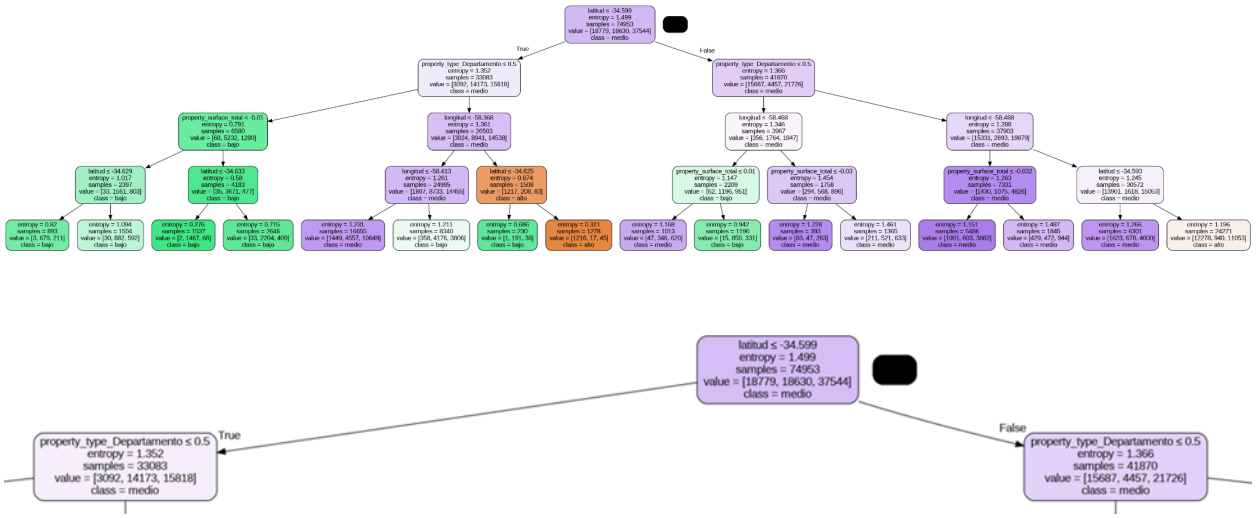
De esta forma evitamos que el árbol sea lo suficientemente profundo y específico para aprenderse totalmente los datos de entrenamiento y ser pobre a la hora de predecir con nuevos datos.

Para esto utilizamos K-folds Cross Validation con 10 iteraciones y 15 folds, dándonos, de esta forma, el mejor score que encontramos para el dataset que formamos.

La métrica elegida para la búsqueda de estos hiperparámetros fue *f1_score* porque, como resulta un balance entre *precision* y *recall*, nos brinda la información necesaria para intentar no obtener falsos negativos/positivos, es decir predecir que una propiedad tendrá un bajo valor cuando tal vez es el caso extremo contrario.

Por lo tanto, los mejores hiperparámetros obtenidos fueron:

{'min_samples_split': 7, 'min_samples_leaf': 1, 'max_depth': 4, 'criterion': 'entropy', 'ccp_alpha': 0.0}



Vemos en el árbol, que el nodo raíz es la latitud siendo este el feature con mayor importancia. Al entrar un nuevo registro para ser clasificado se pregunta si la latitud es menor o igual a -34.599.

En el caso que sea verdadero, pregunta si el tipo de la propiedad no es departamento, si resulta verdadero baja al siguiente nivel donde se verifica la superficie total, en caso contrario pregunta su longitud y sigue el recorrido. Volviendo al nodo raíz, en caso que la latitud sea mayor pregunta nuevamente si el tipo de propiedad no es departamento, se pregunta en el próximo nivel su longitud y sigue haciendo comprobaciones hasta llegar a una hoja en donde termina clasificándose.

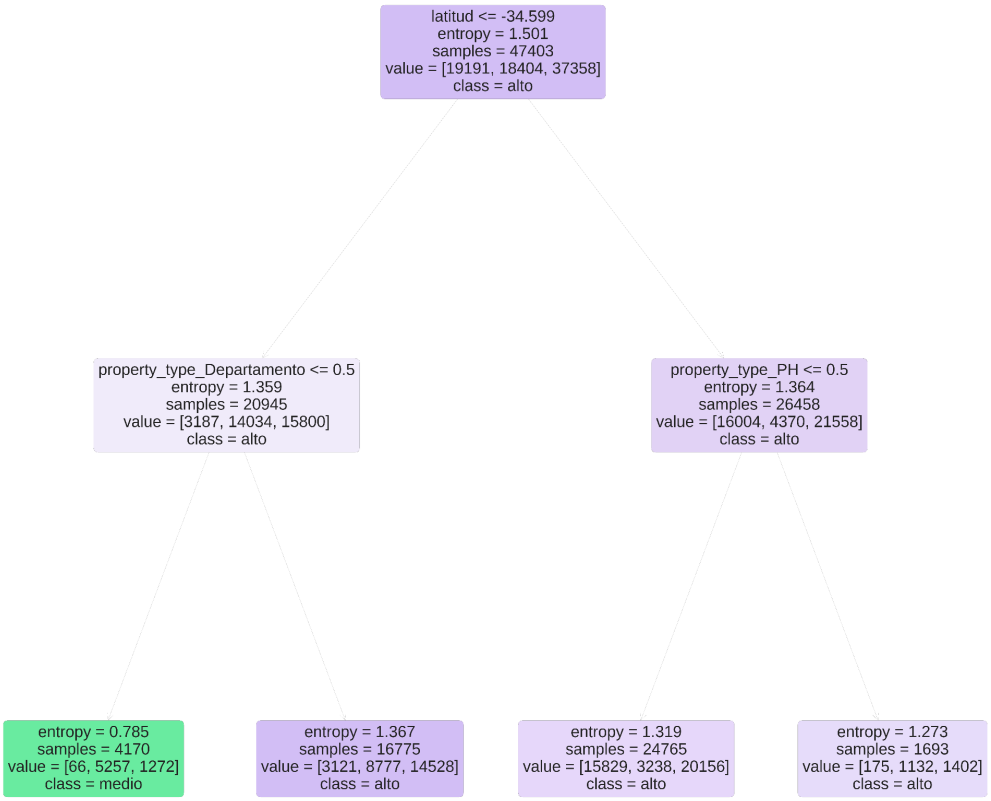
Clasificación por Random Forest

También utilizamos los parámetros **criterion**, **min_samples_leaf**, **min_samples_split**, **ccp_alpha** y **max_depth**. Al igual que anteriormente, utilizando K-fold Cross Validation con 5 folds llegamos a la mejor métrica utilizando, también, *f1_score* para comparar ambos resultados obtenidos con los distintos modelos de clasificación.

Para este modelo, los mejores hiperparámetros encontrados fueron:

{'min_samples_split': 6, 'min_samples_leaf': 7, 'max_depth': 5, 'criterion': 'entropy', 'ccp_alpha': 0.022222222222222223}

Adjuntamos uno de los árboles.



Vemos, esta vez, que el atributo de mayor importancia situado en el nodo raíz verifica la latitud del registro que deseamos clasificar. Si la condición es verdadera, verifica si no es un departamento, en caso verdadero lo clasifica como “medio” y en caso contrario como “alto”. El mismo procedimiento sucede en el otro nodo hasta llegar a una hoja.

Clasificación por SVM

Como modelo a elección implementamos **SVM.** Esta vez para la búsqueda de mejores hiperparámetros utilizamos Grid Search CV que prueba todas las posibles combinaciones. Definimos:

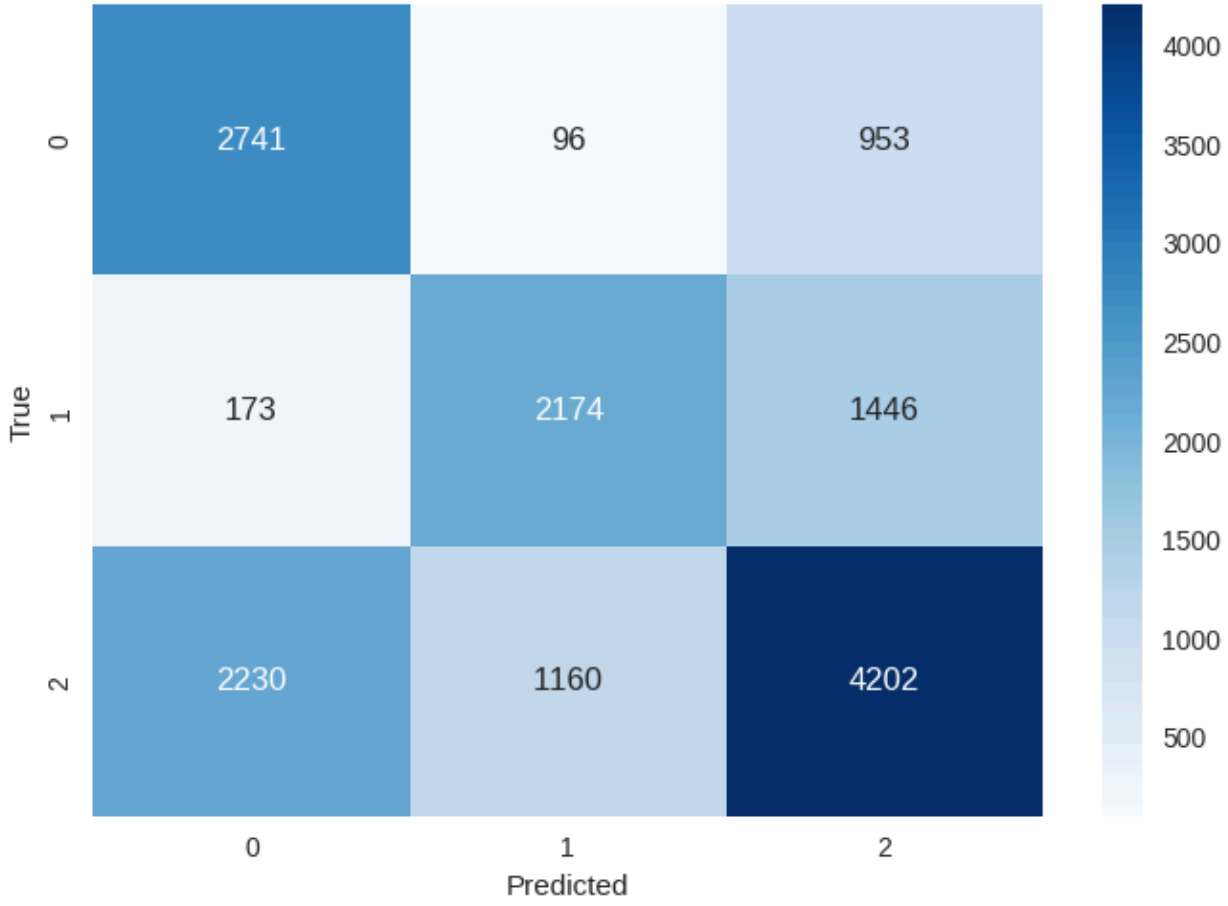
- classifier__C
- -classifier__solver

Obteniendo como los mejores {classifier__C : 100, classifier__solver: 'saga' } con la métrica accuracy.

Cuadro de Resultados

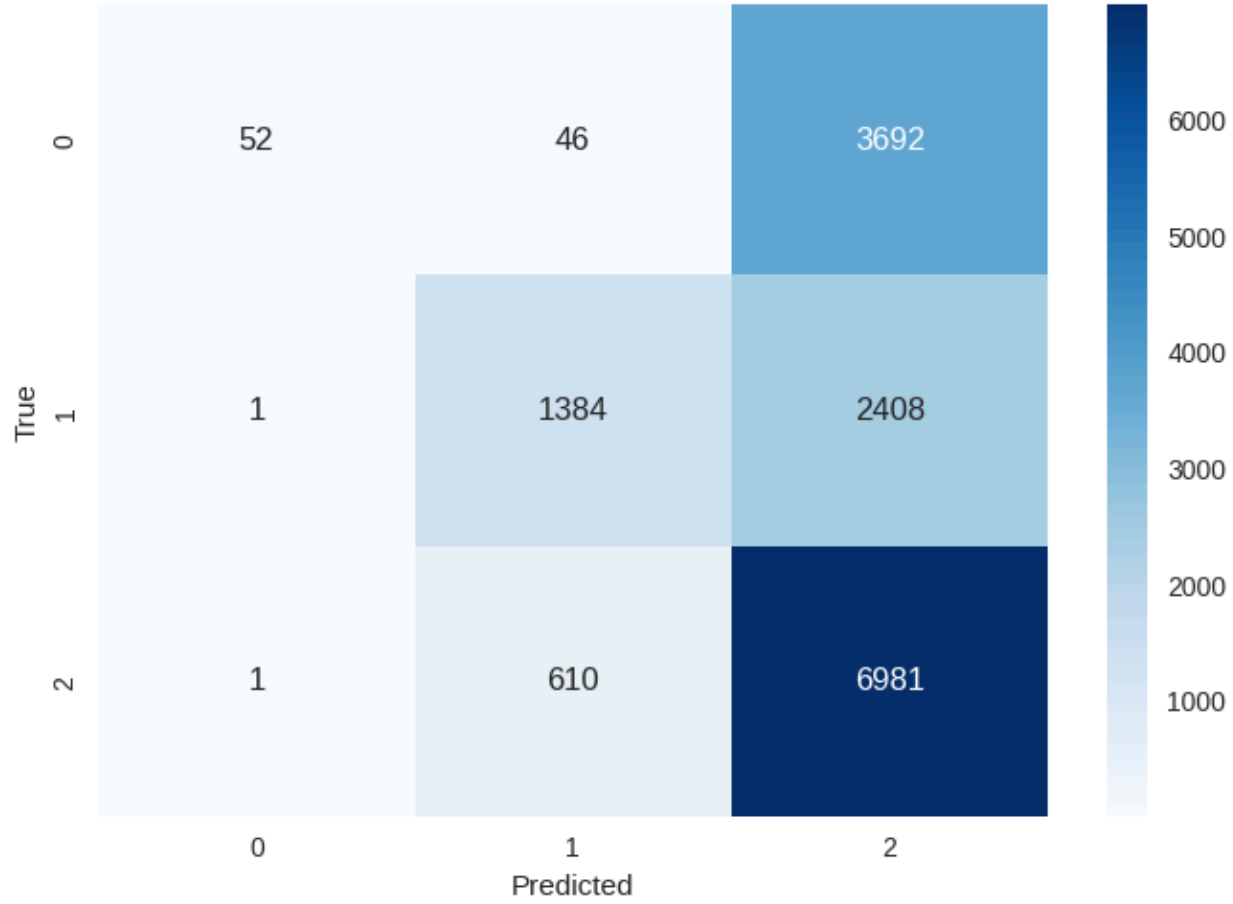
A continuación adjuntamos la matriz de confusión de los tres modelos entrenados.

Árbol de decisión:



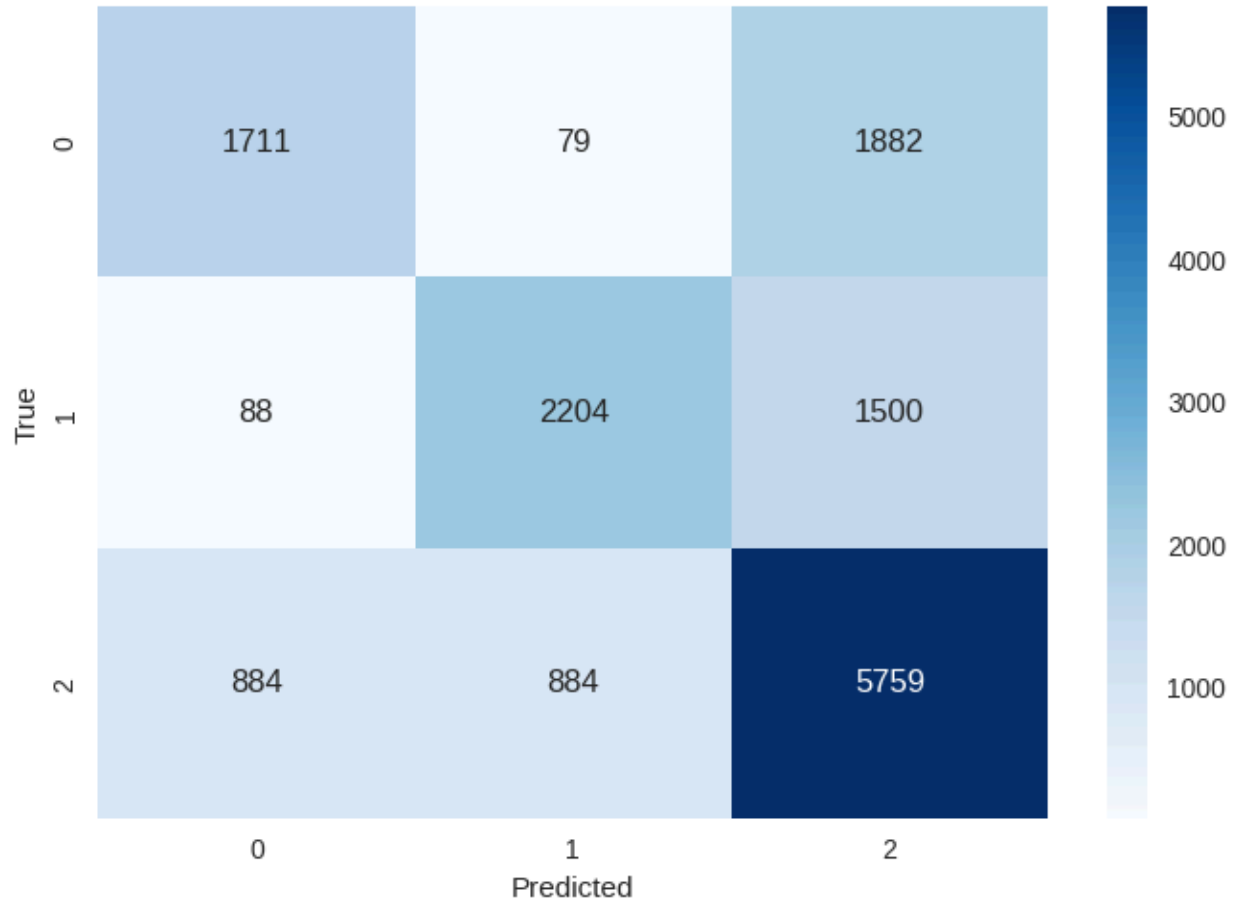
Observamos que para la clase bajo y alto predice muy bien, mientras la clase medio tiene una cantidad considerable de falsos negativos.

Random Forest:



Nuevamente observamos como el modelo también predice bastante bien a la clase bajo mientras que las demás obtienen bastantes falsos negativos.

SVM:



Nuevamente podemos observar que la clase baja la predice bastante bien. Para la clase medio y bajo, en cambio, tiene falsos negativos considerables.

Cuadro de resultados

Modelo	F1-Test	Precision Test*	Recall Test*	Accuracy Test
Arbol de Decision	0.6232	0.63	0.57	0.62
Random Forest	0.5914	0.68	0.36	0.55
SVM	0.6453	0.70	0.58	0.65

* para la clase bajo

La performance en entrenamiento no tuvo cambios significativos, lo que nos indica que no estábamos sobreajustando los datos a la hora de entrenar y testear nuestros modelos.

Elección del modelo

Por un lado, la matriz de confusión del árbol de decisión contiene menos falsos negativos en la predicción de clases en general. Pero consideramos que eso puede deberse en mayor parte a la distribución de los datos, que favoreció al modelo sin afectar en gran medida a sus métricas, como ocurre por ejemplo en el caso de SVM. Este modelo aporta métricas consistentemente mejores, y más certeras para la predicción, reflejado en el valor elevado de F1-score y Precision, si se lo compara a las otras opciones. Decidimos entonces que, de los modelos entrenados, SVM ofrece los mejores resultados para la clasificación.

Regresión

La tarea de esta etapa es predecir el precio de la propiedad a partir de las demás variables del dataset. Se pretende utilizar dos modelos de predicción, más uno a elección. Los modelos esta vez se tratan de **KNN, XGBoost, y Gradient Boost**.

De la misma forma que para la clasificación, sacamos la columna property_price que se intenta predecir, y redujimos la cantidad de columnas del dataset, manteniendo únicamente *latitud, longitud, property_type, property_surface_total, property_surface_covered, property_rooms, y property_bedrooms*.

Además, normalizamos todas las variables numéricas con Z-score, y para el caso de property_type, nuevamente aplicamos One Hot Encoding.

Regresión con KNN

Utilizamos K-fold Cross Validation con 10 folds, el tipo de búsqueda de hiperparámetros fue Random Search, y se buscaron 10 combinaciones. Estos valores fueron con los que alcanzamos el mejor score, viendo que agregar iteraciones no cambiaba significativamente el resultado. La métrica utilizada para optimizarlos fue el Error Cuadrático Medio.

Luego de las iteraciones, los mejores hiperparámetros obtenidos fueron los siguientes:

- **Pesos de los vecinos**: uniformes
- **Cantidad de vecinos para la predicción**: 10
- **Métrica de distancia**: euclídea
- **Algoritmo para calcular distancias**: árbol K-dimensional

Evaluando el rendimiento del modelo con el conjunto de test, se obtuvieron las siguientes métricas:

- El **Error Cuadrático Medio** fue de 17,941,377,896.29485. Representa la media de los cuadrados de los errores, la diferencia entre el valor real y el del modelo. Elevar esta diferencia al cuadrado asegura que los errores positivos y negativos no se cancelen. Sin embargo, es sensible a valores de diferencias grandes, lo cual esperábamos que ocurriera tratándose de predicciones de precios de propiedades.
- El **Error Medio Absoluto** fue de 47,714.66337802682. Esta métrica proporciona una medida más directa de los errores promedio en las predicciones. Esto quiere decir que las predicciones de precios con este modelo tienen, en promedio, un error de $47700, valores dentro de todo aceptables.

Para comparar estos resultados con los obtenidos con el conjunto de entrenamiento, se calculó el score de cada combinación de datos con el coeficiente de determinación **R2**. Es una medida de la calidad del ajuste del modelo a los datos de entrenamiento.

- **R2 conjunto Train**: 0.8092679880900933
- **R2 conjunto Test**: 0.7930632804849669

Como ambos conjuntos presentan variaciones muy similares, esto nos indica que el *overfitting* se pudo mantener bajo control.

Regresión con XGBoost

Utilizamos K-fold Cross Validation con 5 folds, y el tipo de búsqueda de hiperparámetros fue Grid Search. La métrica utilizada para optimizarlos fue el Error Cuadrático Medio.

Los hiperparámetros obtenidos fueron:

- **Magnitud 'lambda'**: 1
- **Tasa de aprendizaje**: 0.1
- **Cantidad de árboles estimadores**: 100
- **Profundidad máxima del árbol**: 3

Evaluando el modelo con los datos del conjunto de test, se obtuvieron las siguientes métricas:

- **Error Cuadrático Medio**: 12,896,555,802.763245
- **Error Medio Absoluto**: 46,226.186128995105

Ambos son valores similares a los obtenidos en el modelo KNN, incluso ligeramente mejores. La métrica en la que se observa una mejora más sustancial es en el Error Cuadrático Medio, indicando que hay una menor cantidad de diferencias grandes entre el valor real y el determinado por el modelo. El Error Medio Absoluto también descendió, aumentando la exactitud de la predicción.

Comparando los resultados del conjunto de entrenamiento con estos, se calcularon los coeficientes de determinación de cada uno de los conjuntos:

- **R2 conjunto Train**: 0.9181216220839679
- **R2 conjunto Test**: 0.851250502258384

La calidad del ajuste a los datos para realizar las predicciones es aún mejor en este modelo, para ambos conjuntos. El modelo puede explicar la variabilidad del precio.

Regresión con Gradient Boost

Utilizamos K-fold Cross Validation con 5 folds, el tipo de búsqueda de hiperparámetros fue Random Search, y se buscaron 10 combinaciones. La métrica utilizada para optimizarlos fue el Error Cuadrático Medio.

Los hiperparámetros obtenidos fueron:

- **Tasa de aprendizaje**: 0.227
- **Cantidad de árboles estimadores**: 145
- **Profundidad máxima del árbol**: 5

Evaluando el modelo con los datos del conjunto de test, se obtuvieron la siguiente métrica:

- **Error Cuadrático Medio**: 4,609,665,002.019385

Es el menor valor que el MSE llegó a tomar en los modelos entrenados, indicando que los valores reales y determinados por el modelo se encuentran más próximos que en los otros modelos.

Comparando los resultados del conjunto de entrenamiento con estos, se calcularon los coeficientes de determinación de cada uno de los conjuntos:

- **R2 conjunto Train**: 0.9479913475900101
- **R2 conjunto Test**: 0.86309409339916

Es el mayor ajuste a los datos de ambos conjuntos hasta ahora. El hecho de que ambas métricas aumenten en simultáneo indica que el modelo está bien entrenado para emitir un juicio razonable sobre nuevos datos desconocidos.

Cuadro de Resultados

Modelo	MSE	RMSE	R2 (Test)
KNN	17941377896.29485	133945.4288	0.79
XGBoost	12896555802.763245	113563.00367	0.85
Gradient Boost	4609665002.019385	67894.51378	0.86

Elección del modelo

En este caso el modelo óptimo en todos los sentidos terminó siendo Gradient Boost, ya que no solo redujo al mínimo posible el error cuadrático medio, que si bien permaneció excesivamente elevado, entendimos que se trata de un promedio entre las diferencias al cuadrado entre los valores reales y las predicciones. Como se trabajaron con los precios de propiedades, las diferencias iban a ser inevitablemente excesivas. La métrica más representativa del trabajo realizado se trata del coeficiente de determinación R2, probando que se puede trabajar con valores que el modelo no conoció durante el entrenamiento, y predecir valores con sentido.
Basándonos en este coeficiente, elegimos Gradient Boost como modelo de regresión.

Conclusiones Finales

Habiendo finalizado con las consignas propuestas, dedicamos este espacio para hacer una retrospectiva sobre lo realizado. El problema en cuestión sin duda tenía una complejidad innata, ya que se trabajó con datos que, si bien fueron recopilados de manera bastante completa por la empresa Properati, no dejaron de ser datos creados por el usuario común. Muchas veces en los avisos de propiedades, algunos parámetros se suelen exagerar, o simplemente omitir, para dar relevancia a determinado aviso. El tratamiento de outliers resultó engorroso de resolver, y nos dejó algunas dudas sobre si las técnicas empleadas para detectarlos y apartarlos fueron las indicadas.

Además, en las tareas de clasificación y construcción del target, no aprovechamos tanto la ingeniería de características en lo que consiste a la creación de nuevos features, que seguramente podrían haber ayudado a obtener clasificaciones, o incluso clusters, con mayor coherencia.

Por otro lado, en las etapas del entrenamiento de los modelos de aprendizaje, podríamos haber buscado combinaciones alternativas de los hiperparámetros para intentar maximizar las métricas de cada método, en lugar de solamente decantarnos por el modelo que entregaba mejores métricas.

Por último, una práctica que tendremos que incorporar para el siguiente trabajo es la división y mejor estructuración del contenido de la notebook, ya que reconocemos que la extensión de esta jugó en contra de muchos procesos de desarrollo.

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
Verónica Leguizamón	Toma de métricas Redacción de informe	6
Henry Maldonado	Modelos de Regresión	6
Augusto Rivera Lofrano	Modelos de Clasificación	6
Alejandro Vargas	Redacción de informe	6