## 

Machine Learning - Ubicar

Laboratorio III

*Profesores:*

*Martin Salaberri*

*Mariana Falco*

*Ignacio Berdiñas*

*Matias Cicilia*

*Alumnos:*

*Maximiliano Adaro*

*Katia Cammisa*

*Pedro Gardeliano*

*Joaquin Iannuzzi*

Índice

[**Resumen del Trabajo**](#_14xrlve081s1) **2**

[**Dataset**](#_3cnu5qvxxx82) **2**

[**Limpieza de Datos**](#_vhd79vgowfni) **3**

[**Nuevas Características**](#_h9vxqqr7dilq) **4**

[**Elección del modelo**](#_q8g2v5x05bo3) **7**

[**Resultados Finales**](#_ws3h6ma57ylj) **8**

[**Posibles Mejoras**](#_1tsp67rysgec) **8**

# **Resumen del Trabajo**

El objetivo central del modelo a implementar, fue desarrollar una herramienta que permita identificar potenciales oportunidades de inversiones, a través de la captura y del análisis de datos de propiedades procesadas con un algoritmo de inteligencia artificial. Se realiza principalmente a través de un dataset de Properati basado en la región de Argentina durante el año 2020/2021. Estos datos pasan por un proceso de limpieza y de transformación.

A su vez, a estos datos les agregamos nuevas características para ser utilizadas como marcadores importantes a la hora de predecir el precio. Principalmente, estas fueron las de los datos geoespaciales de argentina y la característica más relevante es la de “Nearest M2” que nos da a partir de un algoritmo creado por el equipo de Ubicar, el precio por metro cuadrado de una determinada zona. Este algoritmo en la versión final es dinámico, en el sentido de que las zonas no son fijas e iguales, sino que están determinadas por diversos factores que explicaremos más adelante.

Por último, una vez obtenido estos datos con las nuevas características, pasamos el dataset por un modelo predictivo de Inteligencia Artificial ajustando sus parámetros para obtener un modelo óptimo.

# **Dataset**

El dataset principal se basó en un dataset público que ofrece Properati, de todas los inmuebles publicados en su página en el país de Argentina durante el año de 2021. Este cuenta con 894.233 propiedades publicadas. Ahora detallaremos las características dadas de una publicación:

1. **id**: Identificador no nulo, irrepetible.
2. **ad\_type**: Tipo de Anuncio, solo contamos con ‘Propiedad’
3. **start\_date**: Fecha de publicación
4. **end\_date**: Fecha de Término de publicación
5. **created\_on**: Fecha de creación de la propiedad
6. **lat**: Latitud en formato EPSG:4326 - WGS 84
7. **lon**: Longitud en formato EPSG:4326 - WGS 84
8. **l1**:Nivel administrativo 1, país.
9. **l2**:Nivel administrativo 2, provincia.
10. **l3**:Nivel administrativo 3, usualmente ciudad.
11. **l4**:Nivel administrativo 4, usualmente barrio.
12. **l5**:Nivel administrativo 5,
13. **l6**:Nivel administrativo 6,
14. **rooms**: Cantidad de ambientes en formato numérico entero
15. **bedrooms**: Cantidad de cuartos en formato numérico entero
16. **bathrooms**: Cantidad de baños en formato numérico entero
17. **surface\_total**: Cantidad de m² de superficie total
18. **surface\_covered**: Cantidad de m² de superficie cubierta
19. **price**: Precio de la publicación
20. **currency**: Tipo de moneda para el precio de publicación.
21. **price\_period**: Para el caso de alquiler, tiempo de pago ('Mensual', 'Semanal', 'Diario')
22. **title**: Título de la publicación
23. **description**: Descripción de la publicación
24. **property\_type**: Tipo de propiedad ( 'Lote', 'Otro', 'Casa', 'Departamento', 'Oficina', 'Depósito', 'Local comercial', 'PH', 'Cochera', 'Casa de campo')
25. **operation\_type**: Tipo de operación ('Venta', 'Alquiler', 'Alquiler temporal')

# **Limpieza de Datos**

Uno de los puntos fuertes para hacer un buen modelo de inteligencia artificial es tener un dataset limpio. Ahora detallaremos cómo fue todo el proceso de limpieza de datos.

Originalmente, nuestro dataset contaba con 894.233 publicaciones. Lo primero que hicimos fue de las siguientes categorías:

1. price, que no sea nulo
2. currency === “USD”
3. surface\_covered, que no sea nulo
4. l1 === “Argentina”

Con solo estos cuatro filtros, logramos reducir de 894.233 publicaciones a 425.046. La decisión de tomar sólo aquellos con moneda en dólares fue debido a que debida a la situación de 2021 en la Argentina, el valor del Peso fue muy variado. Entonces decidimos dejar aquellas que tengan un valor en dólar que es más fijo y así tener un modelo más preciso.

A su vez, usamos un filtro más, que el ‘operation\_type’ === “Venta”, para tener solo aquellas publicaciones que son para la venta de la casa, sacandonos todas las propiedades que sean de alquiler, ya que la aplicación principal de esta herramienta es para los inversores que quieran comprar inmuebles. Con estos filtros aplicados, reducimos de 894.233 publicaciones a 227.499 propiedades distribuidas en toda la región de Argentina.

También, sacamos aquellas propiedades que estén por encima de los US$8000 el metro cuadrado. Como las publicaciones son hechas manuales y pueden ser precios irreales, este es un buen filtro para eliminar publicaciones erróneas o muy subidas de precio. El valor de US$8000 es sacado como el precio promedio en Argentina de 1 metro cuadrado de la zona más cara de Argentina (Puerto Madero), sumándole $2500 de error.

Al terminar con la Sanitización de datos, removiendo aquellos outliers, terminamos con un total de 191.075 propiedades.

# **Nuevas Características**

Como contamos anteriormente, al dataset le agregamos nuevas features o características. Estas sirven para enriquecer el modelo a la hora de predecir el valor de un inmueble. Una de nuestras principales bases fue el de poder linkear los datos geoespaciales que son públicos en Argentina (*https://www.idera.gob.ar/*) con el dataset ya limpio de Properati. Esto se puede hacer gracias a que ambos datos tienen una posición como referencia, la latitud y la longitud. Los datos geoespaciales que logramos aportar al dataset fueron:

1. 'dAirport': Distancia en metros de la propiedad a un Aeropuerto
2. 'dPort': Distancia en metros de la propiedad a un Puerto
3. 'dTrainStation': Distancia en metros de la propiedad a una Estación de Tren
4. 'dHealthBuilding': Distancia en metros de la propiedad a un Hospital
5. 'dPenitentiary': Distancia en metros de la propiedad a una Cárcel
6. 'dEducation': Distancia en metros de la propiedad a un colegio
7. 'dSubway': Distancia en metros de la propiedad a una estación de subterráneo
8. 'dFireStation': Distancia en metros de la propiedad a una estación de bombero
9. 'dSecureBuilding': Distancia en metros de la propiedad a una comisaría.
10. 'dUniversity': Distancia en metros de la propiedad a una Universidad
11. 'dRailway': Distancia en metros de la propiedad de las vías del tren.
12. 'dIndustrialArea': Distancia en metros de la propiedad a un Área industrial.

A su vez, agregamos a partir de un dataset público de barrios privados en Argentina, tres características nuevas. Este dataset contiene 1074 distintos barrios privados localizados en toda la Argentina.

1. distancia\_barrio\_privado: Distancia en metros de propiedades que no estén en un barrio cerrado pero están a una distancia cercana del mismo
2. nombre\_barrio\_privado: Nombre del barrio privado si es que se encuentra dentro del mismo
3. sin\_cercania\_a\_bp: Dato binario si está o no está cercano a un barrio privado.

Por último, una de las características con más peso fue la llamada ‘Nearest\_M2’. Esta Feature fue una de las más trabajadas en el sentido de que tuvo varias versiones. La idea general de esta característica es de generar a partir de un inmueble, una variable que dependiendo de los inmuebles cercanos, me genere un valor del precio de m² de la respectiva zona. La primera versión constaba en una determinación a partir de un rango fijo. Originalmente fue de 2000m pero después reduciendo a 1000m se logró obtener menor error. La idea es que a partir de una casa que no tenga fijada la variable ‘Nearest\_M2’ (Para evitar el sobreposicionamiento), el algoritmo busque a partir de la posición en el mapa, todas las propiedades que estén a menos de 1000 metros. Al hacer esto, armamos círculos en el mapa con radio de un kilómetro, en el cual cada círculo tiene una determinado precio por metro cuadrado. Asignamos ese precio a todas las casas y seguimos con el resto del dataset, hasta que todas las casas obtengan esa variable.

Para una segunda versión, lo que se quiso hacer es que no sea a partir de un rango fijo, sino que sea por un rango variable, que se agrupen mediante distintos polígonos. Ahora vamos a explicar nuestro algoritmo de casas cercanas basándonos en un rango dinámico.

# **Algoritmo Dinámico Casas Cercanas**

Este algoritmo se basa en ir cortando n-veces un polígono e ir viendo de todos esos cortes cual es el mas optimo y seguir iterando hasta llegar una condición de corte.

Existen 3 variables que el usuario puede cambiar a la hora de aplicar este algoritmo. Para la condición de corte tenemos que elegir: El Área Mínima en kilómetros cuadrados y la cantidad de casas mínimas en un polígono. Por último, la tercera variable es la de la cantidad de cortes que debe aplicar a partir de cada vértice del polígono. Esta variable aumenta grandemente el tiempo de ejecución, pero logra cortes más ‘precisos’ en el sentido de que prueba con más tipos de cortes.

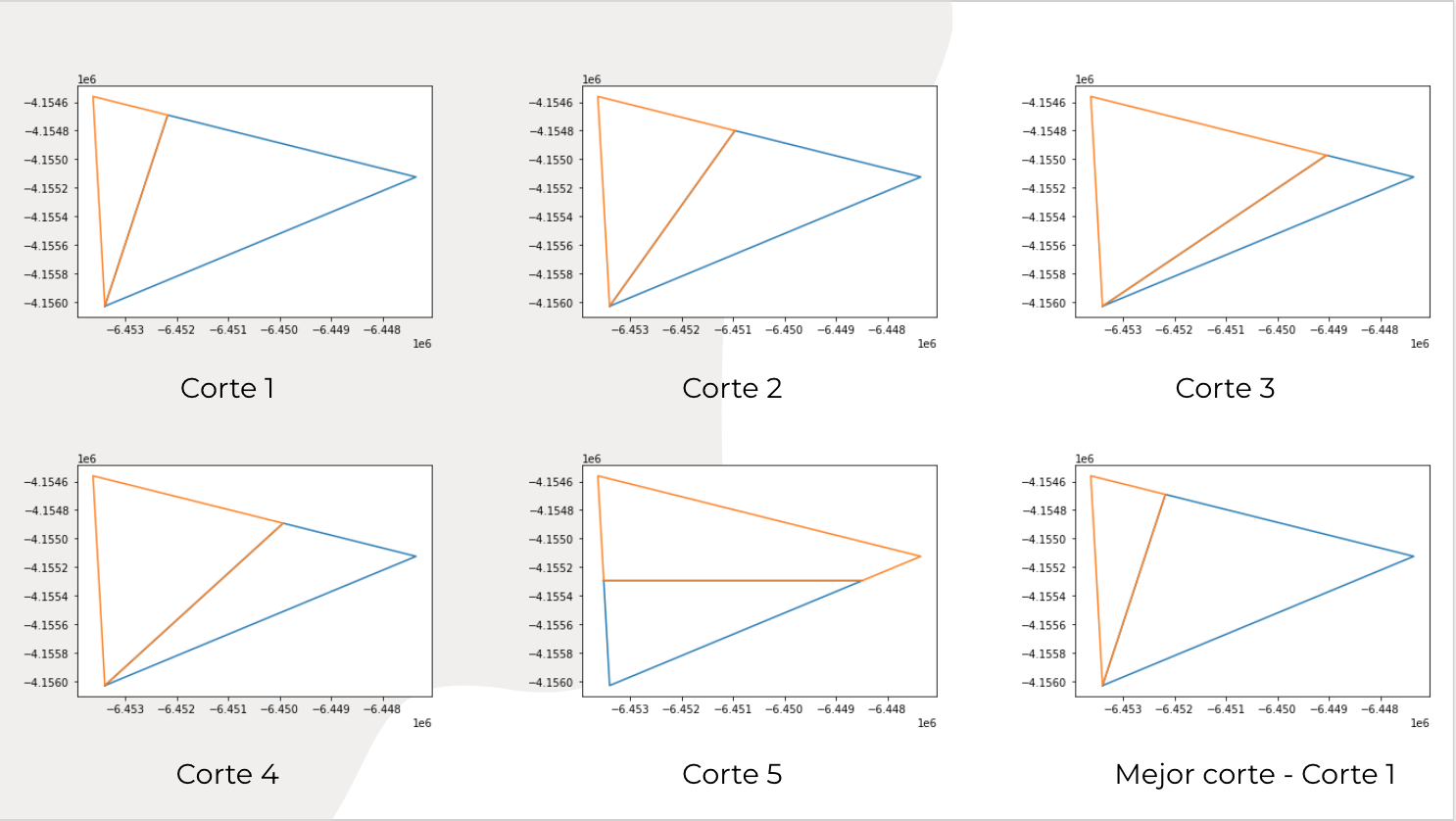
Entonces el proceso detallado es el siguiente:

1. A partir de un polígono:
2. Se separa en 2 o más polígonos recursivamente con un N que fija la cantidad de cortes en cada vértice hasta llegar a una de las siguientes condición de corte:

* Área <= 1.0 km²
* Cantidad de Casas en polígono <= 50

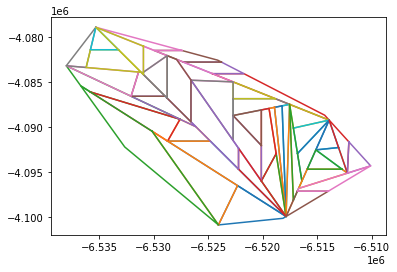
1. De todas las separaciones o splits posibles de ese polígono:
   1. Se buscan todos los splits que cumplan con las condiciones de arriba
   2. De todos los splits, se va tomando el **menor error cuadrático medio** (MAE) comparado con la media del padre para decidir cuál es el mejor. Y así recursivamente hasta terminar con todos los splits creados.

Ahora mostraremos de forma ilustrativa como es que el algoritmo decide.



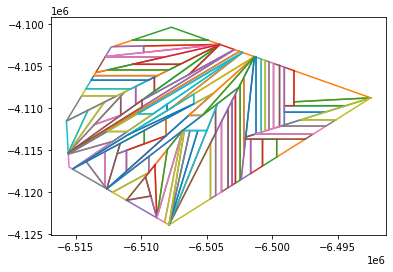
Ejemplo de corte desde un vértice.

En este caso, podemos ver que el algoritmo elige el vértice inferior izquierdo, hizo 5 distintos cortes y en todos ellos calcula el MAE, la cantidad de casas y el área del mismo. Vemos que en la sexta imagen se muestra el mejor tipo de corte que decidió. Como en este caso no llegó a la condición de corte, se va a seguir iterando sobre los dos nuevos polígonos que existen en el corte 1 sucesivamente hasta encontrar una condición de corte.

**Ejemplos del algoritmo finalizado:** 



Polígono de Zona norte, con N=5



Polígono de Capital Federal, con N=10

Podemos ver como del lado derecho queda dividido nuestro polígono original, en el caso de Zona Norte de un polígono pasamos a 47 y en el caso de Capital Federal terminamos con 93 polígonos nuevos. Cada uno de estos polígonos tiene un determinado metro cuadrado.

# **Elección del modelo**

En cuanto a la elección del modelo, lo principal que tiene que tener es que sea lo más preciso posible ya que estamos desarrollando una herramienta que ayuda a inversores a invertir. Inicialmente arrancamos con el modelo de Linear Regression, pasamos por el de Random Forest y finalmente terminamos con el más capaz, el del *extreme gradient boosting*, XGBoost.

La idea de pasar por tres distintos modelos se nos dio para poder entender más a fondo qué es lo que hace cada modelo y evitar el overfitting. Arrancando de un modelo más simple para poder entender más nuestro dataset, hacia uno más complejo y más preciso. A su vez, los modelos de Linear Regression y Random forest, sirvieron como punto de partida para poder explicar al cliente su funcionamiento. Como última versión, pasamos hacia el modelo de XGBoost que también utiliza árboles de regresión como el modelo de Random Forest sumándole otras técnicas de boosteo y técnicas de gradiente. Esto quiere decir que este modelo contiene mejores técnicas en el proceso de entrenamiento como las de regularización mediante Ridge y Lasso, técnicas de *tree-pruning* para evitar el *overfitting* . A su vez cuenta con técnicas de la paralelización y con la capacidad de correr en GPUs que lleva a un menor tiempo de entrenado.

# 

# **Resultados Finales**

Con el modelo final de XGBoost, obtuvimos los siguientes resultados:

| Zona | Cantidad de cortes | Error Mediano | Dentro del 5% | Dentro del 10% | Dentro del 20% |
| --- | --- | --- | --- | --- | --- |
| Argentina | 8 | 10.9% | 27% | 48% | 74% |
| Zona Norte | 5 | 7.351% | 45% | 71% | 91% |
| Capital Federal | 10 | 9.58% | 36% | 60% | 86% |

En este cuadro se detallan 3 tipos de polígonos. Primero el Polígono de toda la argentina, que abarca todas las propiedades disponibles. Segundo y tercero, los polígonos vistos anteriormente en la representación del algoritmo, de Zona Norte y de Capital Federal.

# **Posibles Mejoras**

Algo que podemos notar es que segmentando en zonas, el algoritmo puede reducir notablemente el error. Debido a una insuficiencia de tiempo, no logramos potenciar aún más el modelo. Esto se refiere a poder optimizar ambos los hyperparametros del modelo como también las variables del algoritmo de casas dinámicas. Esto sería ideal hacerlo en algún lugar con alta capacidad de computación, teniendo en cuenta que en una máquina con 4 núcleos, un proceso de 8 cortes, 1.5km² de área mínima y 50 casas como mínimo dura aproximadamente entre 25-30 minutos de *runtime*. Entonces correr en alguna instancia con alta capacidad de computación servirá para disminuir el tiempo de *runtime* en cuanto al algoritmo y disminuir el tiempo de entrenado en cuanto al modelo (Con mejores GPUs).