
# Organización de Datos [75.06/95.58]

## Trabajo Práctico 2

Machine Learning: Publicaciones ZonaProp

2° Cuatrimestre de 2019

Grupo 37 - *Dates*


| Alumno | Padrón |
|---|---|
| Sebastián Ignacio Penna | 98752 |
| Miguel Toscano | 98385 |


*Link repositorio GitHub*: https://github.com/sebapenna/Datos-TP2

# 1. Introducción

En el siguiente informe se describirá, y comentará, sobre el proceso llevado a cabo en el transcurso del segundo trabajo práctico, el cual consistió del desarrollo de algoritmos de machine learning para predecir el valor de las propiedades publicadas en el sitio web ZonaProp.

Para la obtención de estas predicciones no sólo se buscó encontrar los mejores algoritmos (así como sus hiper-parámetros) sino también fue muy relevante el proceso de feature engineering, donde se debió tanto realizar distintos tipos de transformaciones a los datos cómo encontrar nuevos features, derivados de los datos con los cuales se contaba.

Una importante base para decidir cómo transformar o completar los datos del set fue el análisis exploratorio desarrollado en el primer trabajo práctico de la asignatura. En el mismo ya se habían obtenido diversas conclusiones de los datos, muchas de las cuales estaban centradas en el análisis de precios, información clave en el transcurso de éste trabajo.

Es importante resaltar que para la evaluación de los distintos algoritmos se usó la métrica MAE (Mean Absolute Error) para así poder obtener la diferencia entre los precios reales y los predecidos.

Cómo parte del trabajo se formó parte de una competencia dentro del sitio Kaggle, con el objetivo de lograr el mejor modelo predictivo para las publicaciones. A medida que se generarán nuevos resultados, se exportarian los mismos a un archivo csv y luego se realizaria el correspondiente submit a la competencia, de donde se obtenía el verdadero valor de la métrica y marcaba en que algoritmos y/o features debíamos centrarnos.

# 2. Feature Engineering

En esta sección del trabajo se describe cómo se decidió trabajar con los features ya disponibles en el set de datos, tarea que incluyó, generalmente, el siguiente proceso:

1. Análisis de los datos:
    a. Datos nulos;
    b. Distribución;
    c. Importancia frente al precio de una propiedad;
2. Transformación de los datos;
3. Completar datos ausentes;

A su vez, en base a éstos datos y otros obtenidos por nuestra cuenta, se generaron nuevos features que creímos resultan útiles a la hora de generar nuestras predicciones.

En éste caso se contó tanto con un set de train como un set de entrenamiento, por lo que el análisis y transformación de los datos para algunos features no sólo tuvo lugar en el set de entrenamiento, también se debió estudiar el set de train, sacar conclusiones e información del mismo y analizar cómo trabajar con éstos y si se podía complementar el segundo con extracciones del primero, de mayor volumen.

## 2.1. Conversión de tipos

Éste punto no está involucrado estrictamente en el proceso de feature engineering pero si nos pareció importante resaltar cómo se decidió trabajar en éste aspecto.

Al igual que se trabajó en el primer trabajo práctico en primer lugar se convirtió los distintos datos a su tipo necesario (fechas) u óptimo (utilizar el valor

adecuado de int, para así reducir el espacio consumido). De ésta manera sería mucho más ágil el proceso de evaluación de la información y procesamiento de datos.

## 2.2. Tipo de propiedad

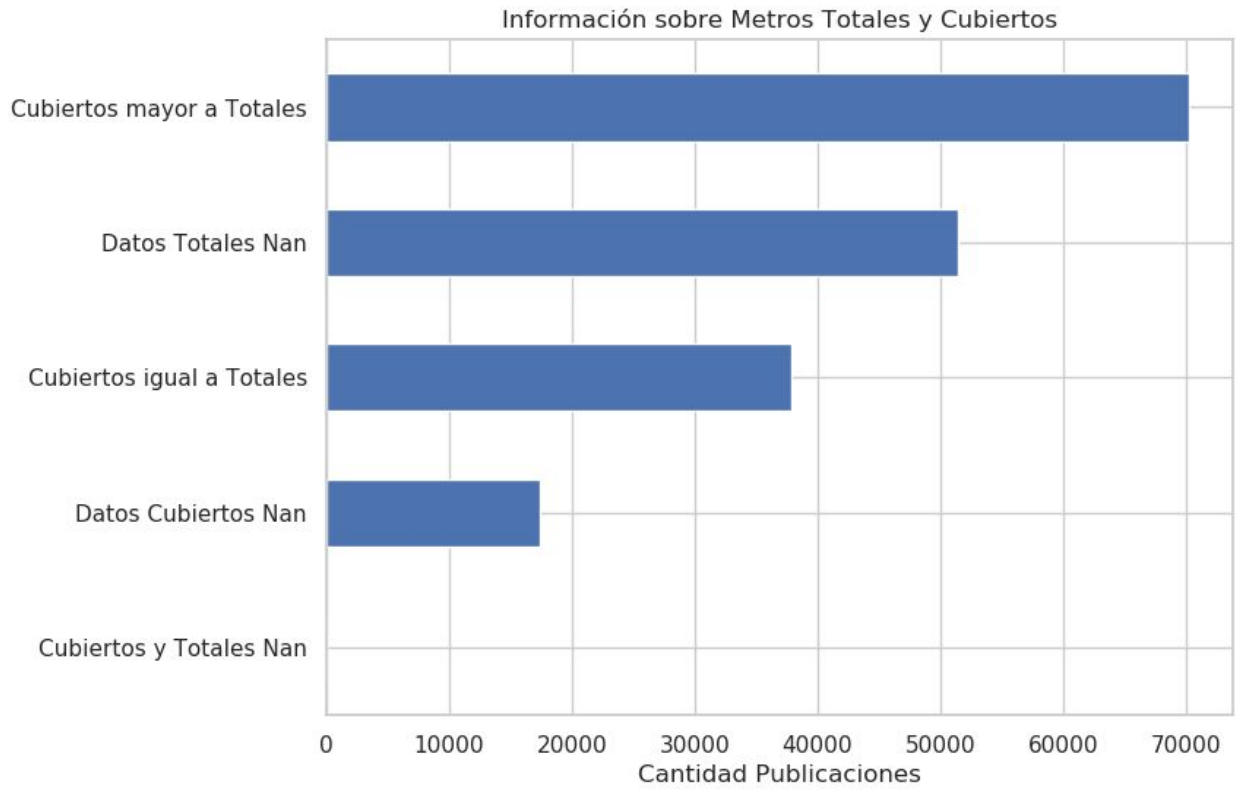
En primer lugar se calculó la cantidad de valores nulos en el tipo de propiedad, obteniendo un valor de nada más que 46 publicaciones. De ésta manera se decidió descartar dichos datos en lugar de completarlos inventando de alguna manera sus valores.

Con los registros restantes se concluyó que la mejor opción (en base a los resultados obtenidos) era la de one hot encoding. Se intentó también las opciones tanto de label encoding y count encoding, pero no lograron optimizar lo obtenido a través de one hot encoding, a pesar de la desventaja de escalar la cantidad de columnas necesarias.

Para el caso del set de test se procedió del mismo modo, pero nos encontramos con que había un tipo de propiedad (Lote) que no existía en el set de entrenamiento.  De ésta manera decidimos descartar la columna generada por one hot encoding para ‘Lote’ (sólo una publicación tenía éste tipo), sabiendo que no se entrenaría un algoritmo en base a ésta propiedad.

## 2.3. Metros cuadrados

Para los metros cuadrados se tienen datos tanto para los metros totales como para los metros cubiertos, por ello se decidió evaluar cómo se comportan los mismos en términos de valores nulos y la relación entre unos y otros. Algo de información se tenía sobre éste aspecto, dado que uno de las secciones del análisis de datos se enfoca en los campos de metros, lo que nos permitió contar con un conocimiento previo.




Un dato positivo con el cual nos encontramos es que en ningún caso se tenía tanto valores cubiertos como totales como nulos, lo que nos permitiría completar el campo faltante en todos los casos.

Un error que se debió corregir es el hecho que los metros cubiertos fuesen mayores a los metros totales, algo que no tiene sentido en una propiedad. Para ello se intercambiaron dichas columnas, de manera de colocar adecuadamente (en base a nuestras ideas) dichos datos.

Realizado éste cambio se procedió a completar los datos nulos, ya que la gran cantidad de datos faltantes impedía descartar estos datos, lo que nos quitaría un gran porcentaje del set y, como luego evaluaremos, son de gran importancia para la predicción del precio. Para esto calculamos la diferencia entre metros totales y metros cubiertos para todas las publicaciones (aquellos registros con valores nulos obtendrían como resultado Nan y por lo tanto no afectaría a la transformación) y luego calculamos tanto el promedio como la media de dicha diferencia por propiedad (evaluando con los algoritmos de machine learning nos encontramos con mejores resultados al utilizar la media). Con éste dato completamos el dato que fuese nulo, sumando la diferencia a los metros cubiertos en caso que los totales

fuesen nulos y restandosela a los totales en el caso que los cubiertos fuesen los nulos.

Para el set de datos se realizó el mismo procedimiento ya que nos encontramos con la misma distribución mostrada en el gráfico previo, pero con una cantidad de datos obviamente menor.

## 2.4. Fecha

Algo que observamos en el primer trabajo fue que los precios se incrementaron a lo largo de los años, más allá de las características de los mismos. Por ello, tanto para el set de entrenamiento como de test, se extrajo del campo conteniendo la fecha de la publicación su año y mes, a partir de los cuales se genero una posición del 1 al 60 (cantidad total de meses) siendo 1 : 2012-01-01 y 60:2016-12-12 y así establecer la franja temporal de la publicación.

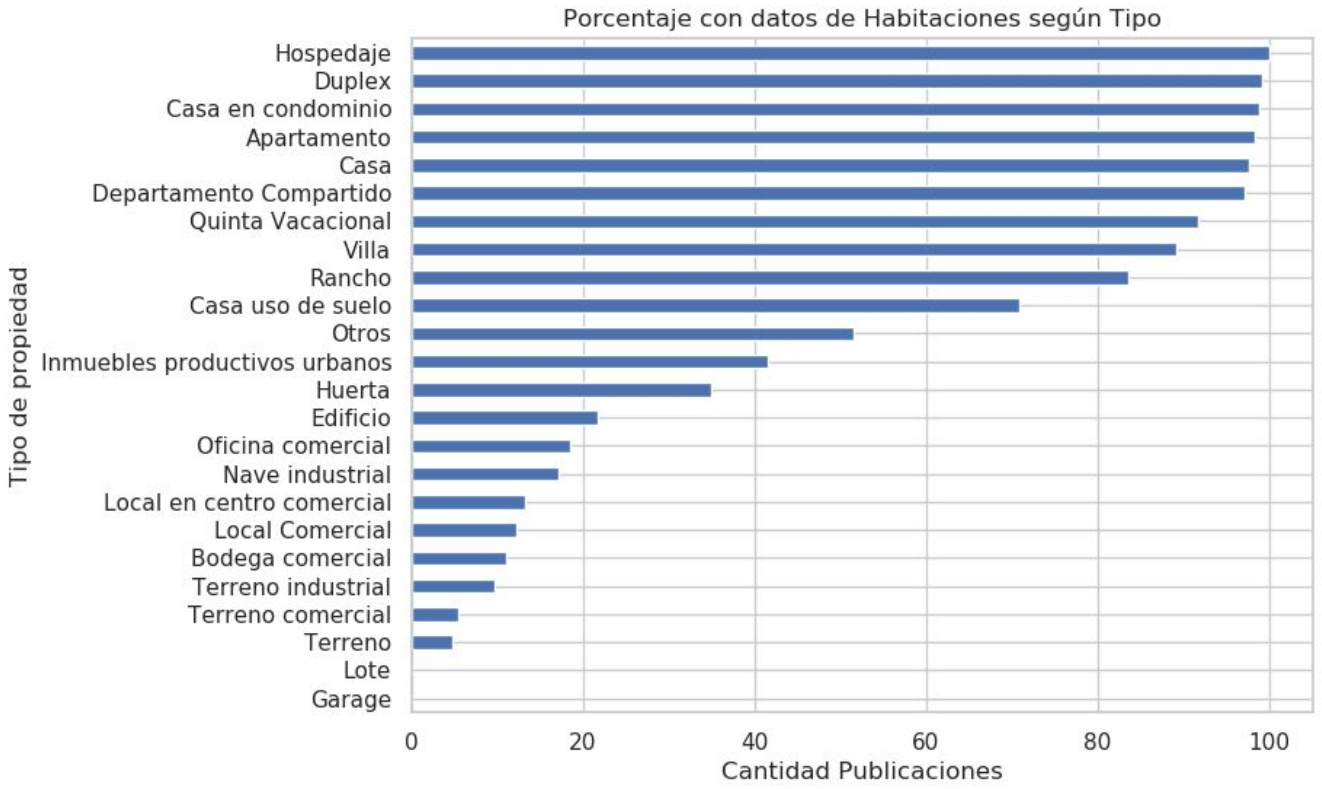
## 2.5. Habitaciones, Baños y Garages

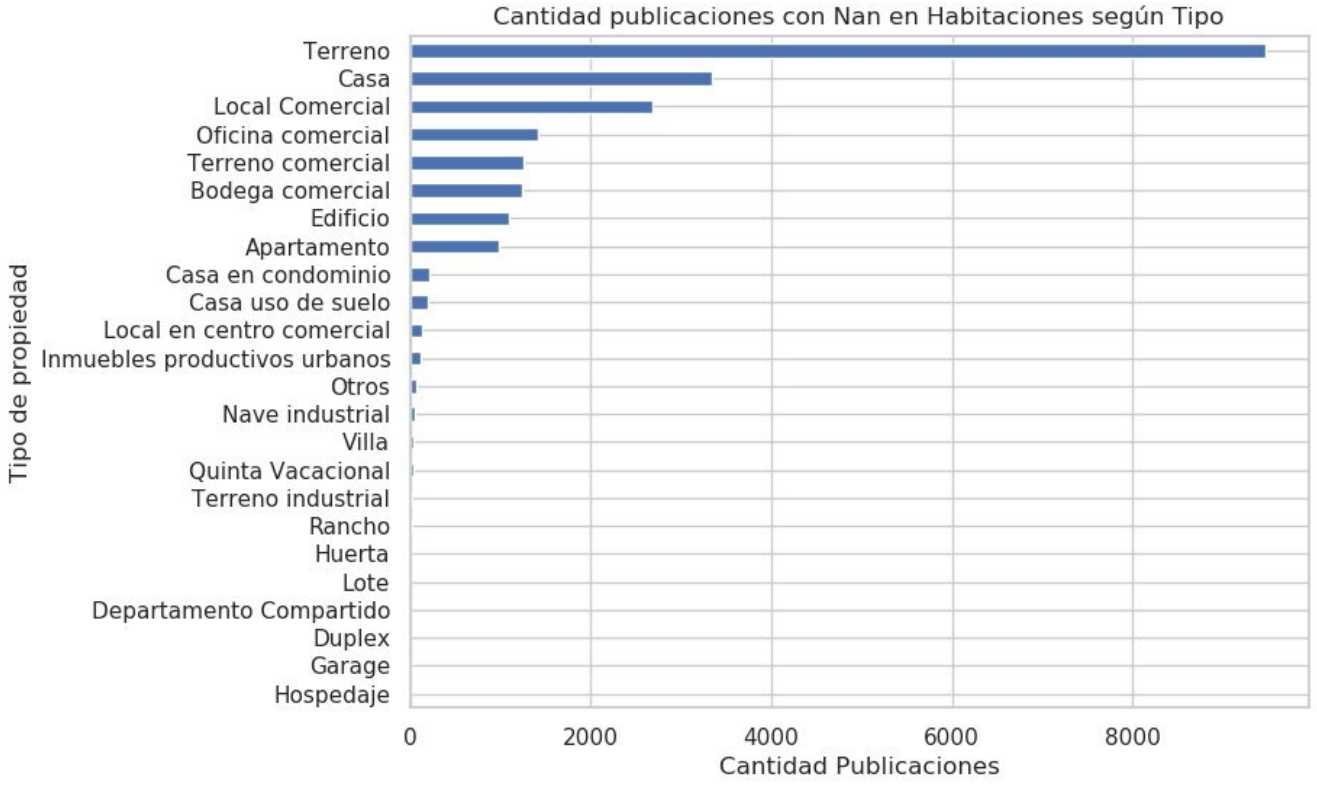
Agrupamos éstos tres features en una sección, dado que el procedimiento llevado a cabo para el completado o descartado de datos fue muy similar para los tres.

En primer lugar se probó descartar los datos que fuesen nulos en alguno de estos campos, lo que descartaba alrededor de un 15% de los datos. Cómo era de esperarse no se obtuvieron buenos resultados con ésta acción, dado que se perdía gran capacidad de extraer información de numerosos registros.

Luego se probó completar los datos manualmente. Para ello se generaron visualizaciones para complementar los datos en cuanto a la cantidad de valores nulos para cada tipo de propiedad (sabiendo que estos valores suelen marcar tendencias en base al tipo) y también del porcentaje del cual si se tiene datos para

las mismas. Como ejemplo se muestran los plots para el caso de las habitaciones en el set de entrenamiento.



A partir de éstos plots se definía cómo se completaban los datos, generalmente llenando sólo aquellos valores para los cuáles se tuviesen más del 50% de los datos, ya que de lo contrario se crearían datos con una base muy escasa, o también en casos donde la cantidad de valores nulos fuese baja. Para completar los datos se buscaron aquellos valores tendencia para cada tipo (uno o más, según el caso) y se seleccionaba un valor random dentro de éstos para completar las distintas publicaciones y así hacerlo más balanceado. De la misma manera se trabajó para los tres features.

Sin embargo, a pesar de obtener leves mejoras, todavía no generaban contribuciones a las predicciones como se esperaba por lo que se decidió a completar los datos utilizando un Simple Imputer tomando como estrategia el valor más frecuente. Para mejorar sus resultados se tomaba en cada fit_transform el tipo de propiedad correspondiente, limitando así su evaluaciona al tipo adecuado.

En el set de test se trabajó de la misma manera, pero en el último caso se debió completar en primer lugar aquellos tipos de propiedad que fuesen nulos, haciéndolo con el valor ‘Casa’ que era el mayoritario y luego también se debieron completar los valores de habitaciones, baños y garages para los Terrenos industriales y los Lotes, ya que no se poseían datos de los mismos. Una vez realizadas dichas transformaciones se utilizó el imputer como en el set de entrenamiento.

## 2.6. Provincia

Comenzamos por el camino más ágil para transformar y testear estos datos como features en un principio a través de one hot encoding. Como consecuencia de ésto obtenemos una gran cantidad de columnas por cada provincia y con el objetivo de reducir las mismas y probar un nuevo feature decidimos transformar cada provincia en regiones, algo que se trabajó y estudió en primer trabajo. De ésta manera calculamos la región para cada registro y luego se realizó nuevamente un

one hot encoding de estos valores categóricos, obteniendo ahora menos columnas y también mejores resultados en nuestros algoritmos.

También se probó utilizar count encoding en lugar de one hot encoding, pero nuevamente se obtuvieron peores resultados del MAE en la predicción generada.

En cuanto a valores nulos, en éste caso nos encontramos que el set de entrenamiento tenía menos de 200 publicaciones sin provincia, y las mismas también carecían de datos tanto de ciudad como de idzona, de los cuales podríamos haber obtenido el valor de dicha provincia. De ésta manera, se decidió descartar los datos nulos dado que completarlos hubiese sido completamente aleatorio y la cantidad de los mismo tenía un impacto muy bajo en el tamaño del set.

Para el set de test se trabajó de la misma manera con la diferencia que en éste caso no podíamos descartar los datos, por lo que nos vimos obligados a completar aleatoriamente (algorítmicamente) el valor de la provincia donde fuese nula (se tenía nuevamente que toda publicación sin provincia tampoco tenia ciudad ni idzona).

## 2.7. Valor del Dólar

En el primer trabajo evaluamos cómo se comportan tanto los precios de las publicaciones como el valor del dólar (en relación al peso mexicano) y ambos mostraron un crecimiento en todo el periodo para el cual se obtenían datos. Por ello nos pareció que un feature que podría influir en la predicción del precio sería el valor del dólar en cada día, dato con el cual también se experimentó en el anterior trabajo obteniendo previamente un set con dichos datos.

Dado que todas las publicaciones (tanto en el set de entrenamiento como de test) tienen dato para la fecha no hubo que hacer transformaciones en los datos, sino tan sólo mergear ambos sets de manera adecuada con tal de obtener éste feature.

## 2.8. Valor Metro Cuadrado

En los siguientes casos se utilizarán los datos del set de entrenamiento para generar los features del set de test, dado que éstos no cuentan con el precio y no se puede calcular el valor del metro cuadrado.

### 2.8.1. Por Tipo

A partir de las transformaciones previas todas las publicaciones contaban con tipo de propiedad, por lo tanto se calculó para cada publicación el valor del metro cuadrado (utilizando los metros totales) en dólares y luego el promedio por el tipo de propiedad. De ésta manera se puede completar toda publicación con su valor del metro cuadrado según su tipo, lo que también nos da una representación del mismo.

También se probó utilizando tanto el valor en pesos mexicanos como la media, pero arrojaron resultados en la métrica algo más bajos que los mencionados en primer lugar.

### 2.8.2. Por Ciudad

En éste caso dentro del set de entrenamiento si nos encontramos con datos ausentes en la ciudad, pero cómo se trataba de tan sólo 200 campos (aproximadamente) se decidió descartar lo mismo y proceder del mismo modo que en el feature último mencionado, pero agrupando por ciudad.

Para el set de test en aquellos casos donde la ciudad fuese nula se completó con la mediana de todos los promedios.

### 2.8.3. Por IdZona

Nuevamente nos encontramos con datos nulos, pero de gran volumen en comparación a la ciudad, por lo tanto hay que completar los mismos para poder utilizar el idzona y no descartar gran parte de los datos. Para ésto se obtienen los distintos idzona para cada ciudad (si es que tienen, si hay valores nulos se descartan) y cada provincia (sólo se usará en caso de que una ciudad no presentará idzona) con los cuales se completarán los id's ausentes, tomando un valor random dentro de la lista que presenten, en caso que haya más de un valor posible. Luego el cálculo del valor de metro cuadrado se realiza de la misma manera que en los casos previos.

## 2.9. Población

Un aspecto que nos resultó interesante para agregar como feature era la población de cada provincia. Para ello se buscó en la web datos sobre la población en cada una de ellas (se completó con datos del 2017) y en base a éstos se generó un nuevo feature a partir de la provincia de cada registro.

Para el set de tests, en el caso que no se tuviera dato sobre la provincia, se tomó como población la mediana de todas las distintas poblaciones.

## 2.10. Largo de Título y Descripción

Otro aspecto que nos pareció podía influir a la hora de la predicción del precio es como está "diseñada" la publicación. Un buen punto de partida sería medir el largo del título y descripción y compararlos a otros.

El feature que generamos seteaba en un 1 (tanto para título como descripción) en caso de que el largo del texto del registro en cuestión fuese mayor al

promedio del correspondiente campo. De la misma manera se trabajó para el set de test.

### IdZona

Al igual que se consiguió el precio del metro por la zona también se decidió completar el idzona en base a la ciudad y/o provincia, donde fuera posible, y así conservar ésta columna, dado que la cantidad de valores nulos era alta pero su importancia suponíamos iba a ser interesante.

### Otros

Otros features son aquellos ya presentes en el set de datos como si tiene o no piscina, SUM, gimnasio, escuela cercana y centro comercial cercano, los cuales no requirieron ninguna transformación particular.

# 3. Algoritmos

## 3.1. KNN

El primer algoritmo que decidimos probar fue KNN, que a pesar de ser un clasificador puede encontrar el valor de una regresión en base a sus k vecinos más cercanos. Cómo era de esperarse éste algoritmo no nos brindó los mejores resultados, pero nos permitió una primera aproximación al problema con una baja cantidad de hiper-parámetros para tunear, haciendo del mismo rápido para probar en un comienzo.

## 3.2. Random Forests

Luego se desarrolló un modelo con éste algoritmo que nos permitió tanto mejorar los valores obtenidos en un comienzo como obtener las primeras métricas sobre qué features eran los verdaderamente óptimos e influyentes a la hora de predecir el precio de una propiedad. Un gran problema que se encontró al utilizar Random Forests era el gran tiempo que requería el entrenamiento de dicho modelo, lo que ralentizaba el proceso de prueba y error de nuestros features.

## 3.3. XGBoost

Una vez que comenzamos a avanzar con el trabajo y random forests dejaba de mostrar mejores en sus valores decidimos comenzar a testear XGBoost, reconocido como uno de los mejores algoritmos de machine Learning. Como se esperaba se comenzaron a obtener mejoras valores para las predicciones y tiempos de ejecución más bajos. Al igual que con Random Forests podíamos obtener un ranking de los features utilizados, lo que nos servía como guía sobre dónde poner el foco de estudio en el problema.

## 3.4. CatBoost

Ya probado XGBoost realizamos un número de pruebas con CatBoost, algoritmo conocido por tener una estructura similar a XGBoost, pero con tiempos de ejecución extremadamente rápidos, lo que lo brinda como una gran herramienta para la prueba de nuevos features. Sin embargo, el puntaje que brindaba era aún menor a XGBoost.

## 3.5. Deep Learning

Para aplicar deep learning, utilizamos la API de Keras corriendo sobre TensorFlow. Hicimos esta elección debido a su popularidad y porque consideramos que está bien documentada.

Para la variedad de modelos que utilizamos, decidimos fijar ciertos hiperparametros para disminuir el espacio de búsqueda de modelos adecuados:

- **Función de activación**: Una elección popular para redes neuronales es la función *Rectified Linear Unit (relu)*.
- **Optimizador**: De nuevo, una elección popular de optimizadores es el optimizador *Adam*

## 3.6. Multi Perceptron

Como primer modelo utilizamos una única capa de 10.000 neuronas. Investigando sobre este tipo de arquitecturas llegamos a la conclusión de que el mismo solo podría ser capaz de aprender a predecir los datos con los cuales se entrena, un ejemplo de overfitting. Esto lo pudimos comprobar al obtener muy buenos resultados sobre el set de entrenamiento pero muy malos sobre el set de test.

## 3.7. Multi layer Perceptron

Al querer comenzar a aplicar estos modelos, elegimos arbitrariamente la cantidad de capas y neuronas por capa a utilizar. De acuerdo a los resultados que fuimos obteniendo, modificamos a mano las dimensiones de la red. Indagando un

poco más en la metodología de cómo buscar arquitecturas adecuadas descubrimos herramientas populares para automatizar dicho proceso. Particularmente usamos la librería *keras-tuner* para encontrar las mejores arquitecturas dentro de un espacio randomizado

## 3.8. Ensamblaje de modelos

Una técnica muy utilizada en Deep learning consiste en generar distintos modelos con distintos tipos de arquitecturas y generar un promedio entre sus predicciones. La idea de esto es que distintos modelos pueden aprender cosas distintas sobre el mismo set de datos. Particularmente decidimos utilizar ambos tipos de arquitecturas previamente mencionados. Haciendo esto obtuvimos mejores resultados que utilizando ambos por separado.

## 3.9. Dropout

Otra técnica que utilizamos para disminuir el *overfitting* en modelos con varias capas, fue la de agregar capas de *Dropout.* Los valores que usamos para las mismas los obtuvimos mediante *RandomSearch.* La implementación de esta técnica mejoró los resultados obtenidos por dichos modelos.

# 4. Selección Hiperparametros y Evaluación

Para la selección de los hiper-parámetros para los distintos algoritmos se trabajó tanto con GridSearch cómo con RandomSearch, según el caso y la cantidad de éstos lo precisara. Una vez que se obtuvieron valores aparentemente óptimos en algunos casos también se intentó mejorar alguno de éstos valores de manera manual a través de ciclos for, por ejemplo, de manera de agilizar el tiempo requerido por los algoritmos antes mencionados.

Para la evaluación de los algoritmos se trabajó con Cross Validation, algoritmo con el cual se obtuvieron valores de métricas realmente buenos (y cercanos a los mostrados en la competencia de Kaggle) una vez que se consiguieron features lo suficientemente buenos y bien transformados para ajustar al modelo. Generalmente se usaron 5 folds para el proceso de Cross Validation.

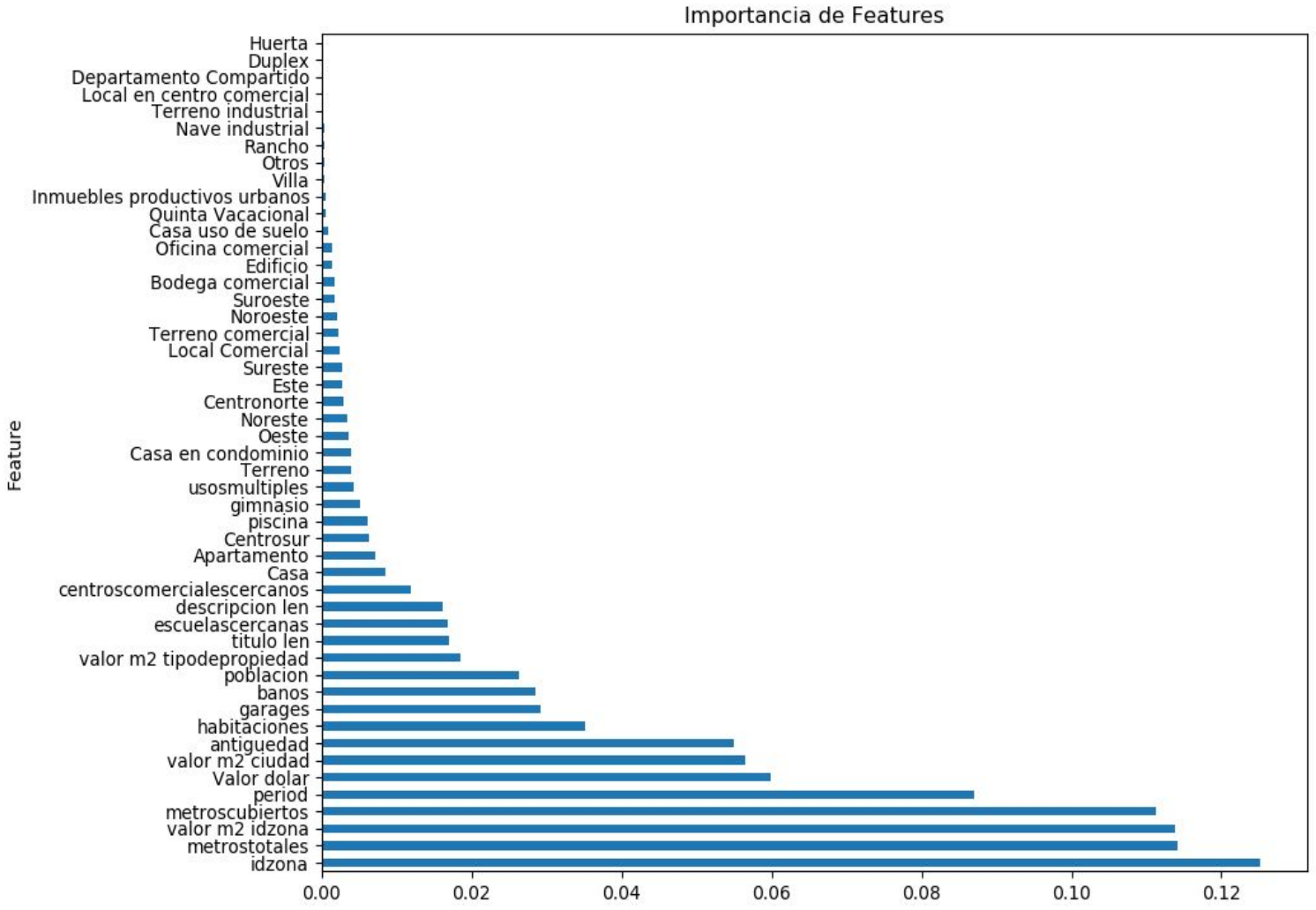
Para evitar posibles relaciones entre los datos en base a su orden en lugar de realizar un train_test_split se utilizó shuffle para aleatorizar los datos y luego utilizarlos en su totalidd dentro del modelo predictivo.

# 5. Mejor Resultado

El mejor resultado obtenido en el transcurso del trabajo se consiguió con XGBoost con los siguientes hiper-parámetros (obtenidos mediante random search):

- min_child_weight=3;
- gamma=0.2;
- learning_rate=0.05;
- max_depth=12;
- subsample=0.7;
- colsample_bytree=0.6;
- n_estimators=400

En cuanto a los features, en base a todo lo explicado y analizado previamente, se obtuvo con éste algoritmo que la importancia de los mismos fue de la siguiente forma:




# 6. Conclusiones

Resuelto el trabajo obtenemos una experiencia en el desarrollo de un modelo de machine learning, particularmente en éste caso un problema de regresión.

Como primera observación se destaca la gran importancia que tiene la transformación y completado correcto de los datos que tenemos en el set así como la generación de nuevos features, ya sea a partir de los mismo o buscando información externamente, como lo hicimos en el caso del dólar o la población. La generación de features se vería potenciada de gran manera si quienes están detrás del proceso de feature engineering tiene un conocimiento sobre el problema en cuestión, los puntos que influyen en la definición del resultado (precio en éste caso) y los posibles agregados que se podrían hacer a los datos ya existentes. En éste

problema resulta algo intuitivo ciertos aspectos básicos que podrían influir en el precio de una propiedad, pero en campos más complejos quizás la experiencia y conocimientos previos sea un requerimiento muy necesario para potenciar los algoritmos predictivos.

Luego se entra en la etapa del desarrollo de algoritmos. En base a nuestra experiencia, una vez que se obtienen buenos features, el proceso de tuneo y búsqueda de los hiper-parámetros óptimos lleva un tiempo prolongado pero la definición de éstos permite influir, aunque sea en lo mínimo del puntaje, la predicción obtenida. Sin duda alguna, utilizar las herramientas de Random Search y Grid Search, al igual que Cross Validation para el testeo del modelo, son claves en éste sentido, por lo que no usarlas presentaría una seria desventaja.

Analizando en particular el problema en cuestión podemos determinar que las publicaciones, es decir las propiedades, dependen de gran manera del id zona que tengan (su barrio) al igual que de los metros que posee, algo bastante lógico. Una primera recomendación que se podría hacer al sitio de ZonaProp sería establecer controles más rigurosos a la hora de setear los datos para una nueva publicación: tanto el idzona, como los metros cubiertos y totales (y su correcto orden), como las habitaciones, baños y garages son features que mostraron ser de vital importancia en nuestros modelos, por lo que información real, con la cual no tendríamos que inventar cierta información para ciertos casos nos brindaría una base verídica aún más fuerte lo que permitiría un modelo predictivo más certero. Como ya se mencionó en muchos casos se tuvieron que seleccionar distintos criterios para completar los datos, así que al quitar del medio éste problema podríamos esperar obtener aún mejores resultados.

Por último, como conclusión general del trabajo, podríamos decir que un buen algoritmo de machine learning está obligatoriamente determinado por un correcto trabajo en todos los puntos del mismo: desde la transformación de datos, completado de nulos, generación de nueva información hasta la selección del algoritmo que mejor ajuste al problema y su respectivos hiper parámetros. Si se deja de lado cualquiera de éstas ramas el conjunto total del mismo fracasará y muy posiblemente encontremos errores en nuestras predicciones, ya sean de overfitting o underfitting, hasta incongruencias en los datos.