FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico N°2
Competencia de Machine Learning

[75.06/95.58] Organización de Datos
Curso 1
Segundo cuatrimestre de 2019

Grupo 39

Padron:	Alumno:
102740	CONTI, Gianfranco
102445	FARETTA, Yanina Belen
100560	GARCIA, Ailen
101807	GIMENEZ, Lorenzo

Link del Repositorio
http://github.com/lzogimenez/orgadatostp1

Grupo en Kaggle
"Salven a los pandas"

Índice

1. Introducción

Como continuación del análisis exploratorio realizado en el Trabajo Práctico N°1 ésta segunda parte nos desafió a predecir el precio de un set de propiedades a partir de técnicas de Machine Learning aprendidas en el curso.

Nuevamente el set original por si solo contaba con las siguientes features:

- Titulo, descripción y fecha de publicación
- Dirección, ciudad, provincia, idzona, latitud y longitud
- Tipo de propiedad y cantidad de habitaciones, garages y baños
- Antigüedad, metros cubiertos, metros totales y precio
- Indicadores de gimnasio, salón de usos multiples, piscina y escuelas y centros comerciales cercanos

El objetivo de esta práctica es conocer y probar distintos algoritmos, para realizar feature engineering (donde puede ayudar el análisis exploratorio realizado en la práctica anterior), cometer errores e ir evolucionando hasta llegar a una buena predicción.

2. Desarrollo y Mejoras

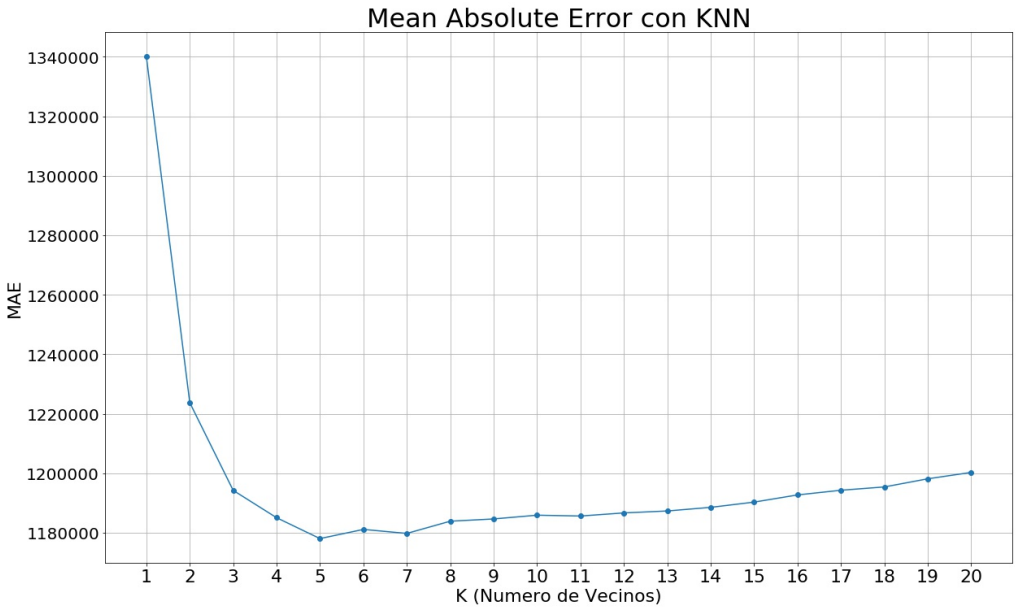
2.1. Modelos Utilizados

2.1.1. KNN - 'Por algo se empieza'

Siendo nuestra primer experiencia implementando un algoritmo de predicción, nuestra primer elección de algoritmo fue una bastante intuitiva y (al menos en éstas instancias) todavía no nos interesaban las posibles restricciones que podía conllevar.

Luego aprenderíamos que esas restricciones serían determinantes de si un método nos es o no útil, para éste modelo, debimos quitar todas las filas que contengan valores NaN y todas las columnas que no fueran numéricas

Su implementación fue rápida una vez descargadas las librerías y tras ver que funcionaba fuimos un poco más lejos e hicimos una optimización sobre K, con el que se obtuvo el siguiente gráfico:



Se obtuvo el mínimo error entre los valores probados para K=5, y en la corrida correspondiente se obtuvo un MAE de 1.178.075,78, y éste primer resultado puede parecer trivial pero lejos de eso nos dió dos de las pautas más importantes para el desarrollo de éste trabajo práctico:

- Complejizar el modelo ($K >> 1$) no siempre implica tener una mejor predicción, porque podemos encontrarnos frente un caso de UNDERFITTING
- Un resultado muy bueno en el train ($K = 1$) no tiene por qué implicar un buen resultado en el test, porque podemos encontrarnos frente un caso de OVERFITTING

NOTA: Sobre la importancia de las features que se están descartando y su posible codificación se hablará en la sección correspondiente, ésta es solo una sección que menciona los modelos utilizados

2.1.2. XGBoost - 'Dijo y me enamoró'

Como su nombre lo indica es un modelo que se basa en el concepto de boosting, donde básicamente la predicción se construye mediante la suma de los resultados de varios árboles (estimadores) que van a intentar corregir el error cometido por el árbol anterior.

No por nada es el algoritmo estado del arte en clasificación y varias competencias son ganadas con modelos de ensamble cuya componente principal es XGBoost.

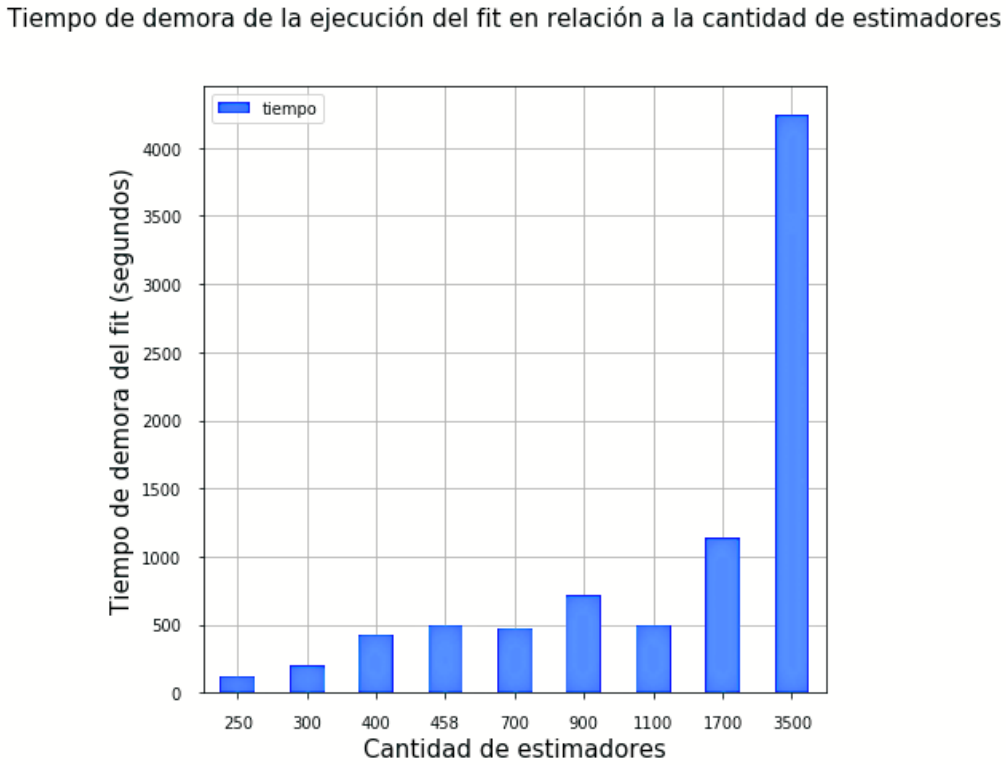
¿Un modelo que es estado del arte, suele ganar las competencias y encima nos permitía entrenar con un set que contenga NaN? - Poco le costó a XGB llamar nuestra atención.

Nuevamente, debíamos dejar fuera a las columnas no numéricas, pero incluso con todos sus hiperparámetros por defecto, XGBoost nos consiguió un MAE de 822.878,42

Sobre este prometedor primer resultado y despues de unas horas de investigación por internet, nos sentimos listos para efectuar nuestra primer búsqueda profunda de hiperparámetros.

Fueron efectuadas con Cross Validation búsquedas Grid y Random y sin modificar el set se logró un MAE de 678.932,19

Un detalle no menor es que correr búsquedas, o fits, sobre XGB fue mucho más costoso que en KNN, de manera que no fue sencillo encarar las búsquedas sin que nuestras computadoras quedaran colgadas por mucho tiempo, de hecho es interesante ver cómo evoluciona el tiempo que demora hacer un fit para las distintas versiones implementadas del modelo (en particular para este análisis de tiempo ponemos foco en la cantidad de estimadores)



El siguiente paso fue empezar a trabajar con las features, pero ese registro estará presente en la sección correspondiente.

2.1.3. Random Forest - 'No espero nada de ti y aun así me decepcionas'

Random Forest es un método de machine learning basado en una técnica de ensamble llamada *bagging*, el método arma varios árboles de decisión construyendo un "bosque"de árboles y, si se lo usa como regresor, realiza el promedio de la predicción de todos sus árboles para obtener el resultado final.

La primera vez que se armó dicho modelo con los features que utilizabamos en ese entonces, obtuvimos un MAE de aproximadamente 0,826215, por lo que pensamos que era demasiado bueno, ya que todos los otros modelos daban valores cercanos a los 700.000, pero claramente dicho resultado se debía a un error de implementación: habíamos utilizado un método del primer trabajo para que los gráficos fueran más fáciles de leer sin percatarnos que éste dividía el precio. Entonces, verdaderamente ese MAE representaba unos 826.215, y demostraba que el modelo básico no era mejor que los probados anteriormente.

Este modelo se armó en paralelo con XGBoost para ver si lográbamos de alguna forma obtener un mejor resultado, a pesar de las restricciones que presentaba Random Forest (no aceptaba NaN, por lo que tuvimos que, o bien desechar todos los NaN, o bien aportarles algún valor para que el modelo los tome (se probaron ambas cosas y obtuvimos un mejor MAE con la segunda pero igualmente seguía muy lejano al de XGBoost).

Luego de pasar varios días intentando mejorar el modelo y de tunear sus hiperparámetros con random search, obtuvimos una mejora respecto al MAE inicial, logrando uno de 811.430. Finalmente concluímos que la diferencia de MAE no iba a ser fácil de solucionar y que éste modelo no era una buena opción para reemplazar al probado XGBoost, sin embargo, continuamos usándolo y comenzamos a crear nuevas features intentando acortar la diferencia entre los modelos.

2.1.4. Regresión Lineal - 'Otro muerde el polvo'

Tras Random Forest intentamos con un nuevo modelo: regresor lineal. Primero intentamos armar una regresión lineal multivariable para ver qué ocurría si usábamos todas las features pero, para sorpresa de pocos, dio un MAE que dejaba bastante que desear frente al del resto de modelos, el resultado incial fue de 1.222.891.

Posteriormente, intentamos hacer una regresión utilizando solamente una variable: metros cubiertos. Pero el resultado en este caso fue un MAE de 1.294.707 (incluso peor que con la regresión multivariable), por lo que nuevamente frente a los resultados de los otros modelos decidimos no continuar con éste y seguir buscando alguno que iguale o supere a los modelos ya probados.

2.1.5. Redes Neuronales - 'Al menos lo intentaste'

A día de hoy al hablar de Machine Learning, es difícil no asociar la idea con redes neuronales, por lo que para nosotros fue natural el querer implementar una, sin embargo esto resultó ser bastante problemático.

Lo primero fue buscar una implementación. Antes de utilizar alguna librería como $TensorFlow$ o $PyTorch$ (que dan más control sobre las operaciones que se realizan en cada capa) decidimos utilizar la api que proporciona $Sklearn$ para hacer un primer prototipo. Luego de tocar varios parámetros, y rellenar los NaNs con el promedio de cada categoría, el mejor MAE que obtuvimos fue de 1.597.524.

Los valores obtenidos no eran los deseados, dado que al momento de probar éste modelo ya contábamos con una versión muy optimizada de XGBoost que nos daba excelentes resultados, de manera que por cuestiones de tiempo decidimos abandonar este modelo.

2.1.6. LightGBM - 'Me hacés acordar a alguien pero no se a quien'

LightGBM es un modelo basado en *boosting* al igual que XGBoost con la no menor diferencia que el tiempo que requiere para entrenarse es bastante menor al de XGB, pese a que ambos se basan en el *boosting*. Como con XGBoost logramos un buen MAE con éste también esperábamos tener buenos resultados.

Y así fue, el modelo casi por defecto nos daba un resultado similar al de XGBoost sin mejoras. Así que simplemente hicimos una serie de búsquedas para optimizar los hiperparámetros y se utiliaron los mismos features que utilizamos para el modelo de XGBoost, logrando bajar el MAE hasta 488.422, con el modelo entrenándose en aproximadamente 35m. Con lo que teniendo ahora dos modelos muy buenos decidimos encarar un ensamble entre ambos para intentar mejorar aún más las perdiciones.

2.1.7. Ensamble - 'Hacen re linda pareja ustedes dos'

Uno de los últimos recursos empleados, fue efectuar un ensamble entre nuestros dos modelos de mejor predicción hasta la fecha, éstos son XGB y LightGBM (ambos algoritmos de Boosting).

Utilizamos dos tipos de ensamble:

- Multiplicar cada predicción por una determinada fracción para sacar un promedio.
- Utilizamos una biblioteca llamada *mlens*.

Para el primer tipo de ensamble se ponderó con un valor de 0.7 para XGB y un valor de 0.3 para LightGBM logrando que las predicciones sean ligeramente mejores, sin embargo, esperábamos obtener aún mejores resultados con *mlens* pero se obtuvieron resultados incluso peores a los MAE de cada modelo de forma independiente.

2.2. Feature Engineering

2.2.1. Reinventando la rueda

En esta parte queremos documentar cómo fuimos evolucionando en los criterios a tomar a la hora de integrar nuevas features o reacondicionar features existentes.

Como mencionamos en los desarrollos de los modelos, el dataset contaba con columnas no numéricas y éstas inicialmente fueron dropeadas, nosotros, sin ningún tipo de conocimiento de feature engineering per sé y previo a la clase de éste tema, nos hicimos una serie de preguntas que nos hicieron ir creando columnas de distinta calidad.

- ¿Si a cada valor de las categóricas le ponemos un valor numérico entero arbitrario?

  Nuestro primer intento, le permitíamos a nuestros modelos usar mas información. Fue implementado tan simple como suena, por ejemplo para tipo de propiedad 'Casa' se escribía '1' para 'Apartamento' correspondería '2' y así...

- ¿Si a cada uno de esos valores enteros, le damos un orden?

  Ahora, planteábamos que esos números estén asociados al precio promedio de esa categoría en el set de train, de manera que la propiedad cuyo tipo tenga el promedio más barato tendría el valor '1', el segundo tipo más barato tendría el valor '2' y así... Nuevamente fue implementado como una gran tira de ifs para las distintas categorías

- ¿Si en lugar de usar valores enteros, ponemos algún valor que cuantifique la diferencia existente entre el promedio de una categoría y la otra?

  Ésta sería nuestra última implementación antes de la clase de Feature Engineering, la idea era tomar el promedio de la categoría y normalizar los valores obtenidos.

Esta subsección lleva el nombre 'Reinventar la Rueda' porque no eramos conscientes de que todas estas cosas ya existían, en particular la segunda era muy similar a $OrdinalEncoding$ y la última era prácticamente $MeanTargetEncoding$, que una vez que tuvimos la clase de Feature Engineering, supimos explotar mejor,

2.2.2. Mean Target Encoding y otras maravillas numéricas del mundo

Conociendo ahora las distintas técnicas 'famosas' encaramos la creación de distintas columnas numéricas que paulatinamente fueron mejorando los scores obtenidos, destacamos en ésta sección las **numéricas** que mejor score poseen:

- MEAN TARGET ENCODING SOBRE PRECIO
  - `tipo_de_propiedad_mean`
  - `provincia_mean`
  - `ciudad_mean`
  - `idzona_mean`
- ESTANDARIZACIÓN
  - `metros_cubiertos`
  - `metros_descubiertos`
  - `metros_totales`
- TEMPORALES
  - `año`
  - `mes`

2.2.3. Nunca subestimes a una función de Split

Más de una vez nos encontramos con que el MAE calculado localmente sobre nuestro set de test propio tenía una gran diferencia con el scoring en Kaggle de las predicciones generadas por ese mismo modelo. Esto nos llevó a detectar distintos errores en el orden y forma en que calculábamos los features:

- Tamaño del set de test

  Nosotros trabajamos con una función de split que separaba al dataset original *train.csv* en 4 partes x_train y_train x_test y_test, los primeros dos se utilizaban para entrenar al modelo y para las optimizaciones previas (mediante cross validation) y los últimos dos nos permitían estimar el MAE cometido por nuestro modelo.

  Ahora bien, inicialmente estábamos tomando un 20 % del dataset y aunque en algunos casos eso puede ser suficiente, lo ideal sería que el test local, sea de magnitud similar al que se está testeando en Kaggle, por lo tanto nuestra primer modificación a la función de split fue que tome un porcentaje mas grande del dataset original para el test.

- Evitar filtrar información

  Otro error del que aprendimos fue que inicialmente calculábamos los promedios de los mean target encoding sobre todo el set, generábamos las columnas y luego efectuábamos el split.

  Esto está **MAL** porque está utilizando los valores precio de las filas que luego del split corresponden a x_test y éstas deben ser tratadas como si y_test (o sea el precio) no existiera para efectuar las predicciones

  Esta observación conllevó grandes cambios en la función de split y la creación de una función que se encargue de 'mapear' los means calculados solo sobre x_train hacia y_train (que es la misma que se usará para generar los means de *test.csv*)

2.2.4. Análisis sobre los campos de texto

Para generar columnas sobre los campos de descripción y título, se crearon funciones que a través de splits y counts determinaban si **palabras** o **frases** pertenecían a éstos campos.

Para ello los campos fueron 'normalizados':

- A todos los NaN se les dio un '.' como valor.
- Todas las letras se escribieron como minúscula
- Se reemplazaron los puntos y comas por espacios

La función $contar\_palabras$ trabajaba sobre los campos spliteados y la función $contar\_frases$ trabajaba sobre los campos normalizados a secas.

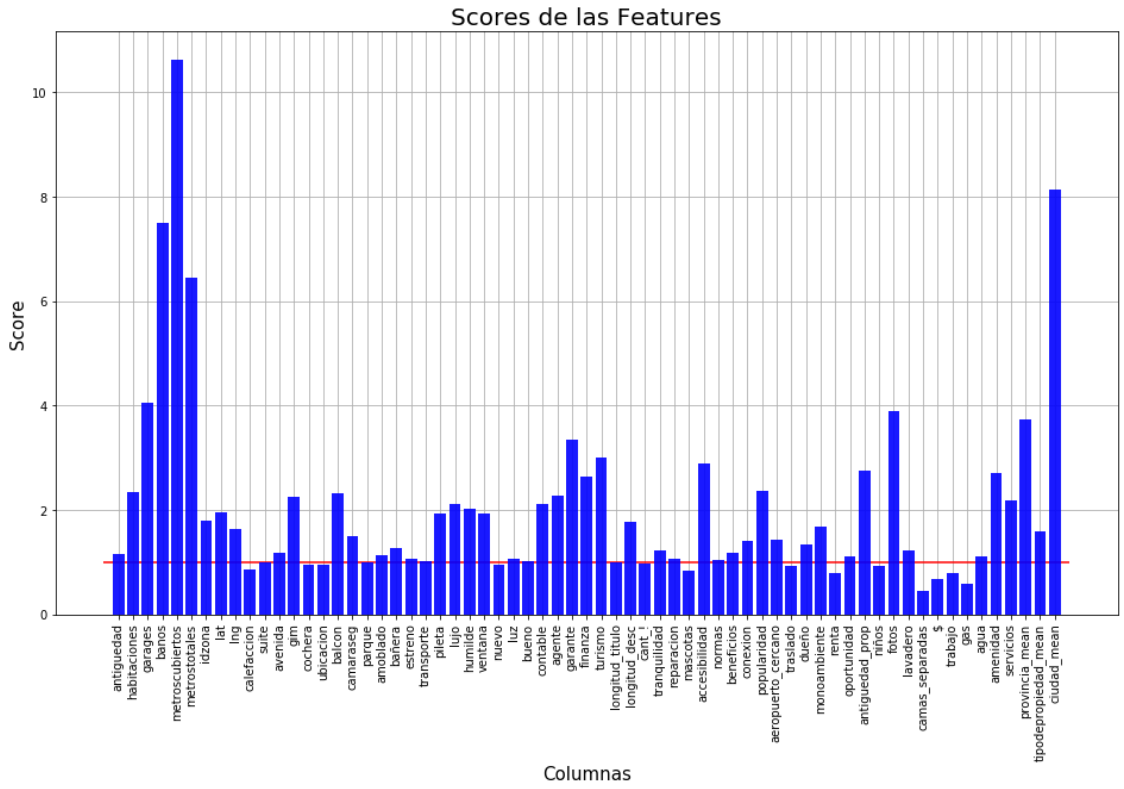
Así fueron generadas columnas como la cantidad de veces que se mencionaba la palabra 'balcón', 'cochera' y 'bañera'. La razón de incluir ciertos features de palabras mucho mas interesantes son presentados en la sección 'Encuesta a Inmobiliarias y Brokers de México',

2.2.5. Feature Selection

Una vez alcanzado un número considerable de features, era necesario comenzar a identificar aquellas features que aportaban buena información al modelo, evitando aquellas que hiciesen solo "ruido" o repitan información ya proporcionada por otras.

Para obtener los scores correspondientes a cada feature se utilizó en primera instancia $SelectKBest$, que calificaba de forma **univariante** a las distintas columnas presentes tomando $K = \#columnas$ pero tenía ciertas limitaciones, en particular, no podía trabajar con columnas que tuvieras valores nulos, de manera que el score se calculaba sobre un dataset más pequeño.

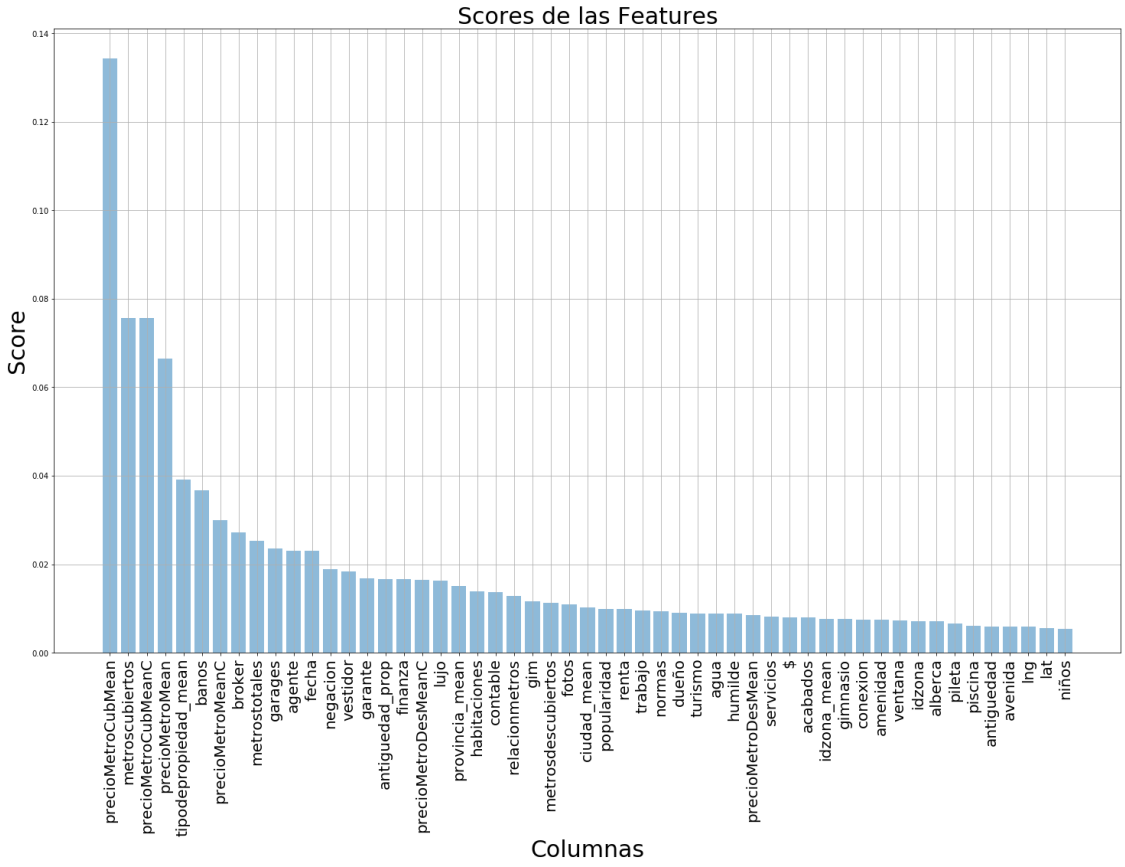
Ejemplo de plot obtenido para el score por SelectKBest



Este approach fue bueno inicialmente, porque nos permitía tener una noción de que tan valiosa podía llegar a ser cada feature, sin embargo, con el correr del tiempo y la adición de muchas features, era complicado hacer un buen seguimiento y no podíamos saber realmente cuáles ponderaba más el modelo a la hora de entrenarse.

Luego, optamos por utilizar el parámetro *feature_importances* propio de XGBoost, que nos daba la pauta de qué tan determinística resulta ser cada una de las columnas al momento de construir los estimadores y esos puntajes **no son univariantes**, si bien éste approach era más fiel a lo que usaba el modelo, era muy sensible a eliminar o agregar columna y eso hizo bastante más tedioso el cambiar features para ver como reaccionaba el modelo.

Ejemplo de plot obtenido para el score por Feature Importance de XGB



3. Encuesta a Inmobiliarias y Brokers de México

3.1. Motivación

¿Puede definir usted qué es la *plusvalia* de un inmueble? ¿Sabía que los *acabados* de una propiedad pueden ser de gran decisión a la hora de estimar su precio? ¿Alguna vez tuvo contacto con un *BrokerInmobiliario*? Hasta hace unas semanas, nosotros tampoco.

Ésta idea se originó durante la creación de features a partir de los campos de texto, particularmente cuando se nos ocurrió señalar aquellas propiedades que contaban con un *ascensor* ¿Suena lógico, no? Una propiedad con ascensor a priori podría ser mas cara que una similar que solo cuente con escaleras, sin embargo al realizar la búsqueda el porcentaje de descripciones que mencionaban *ascensor* en su descripción era menos que el 0.1 %.

¿Cual fue nuestro error? Hasta ese entonces no estábamos considerando la **barrera del lenguaje** (en México no se dice *ascensor*, sino *elevador*, y con esa corrección el porcentaje de descripciones escaló rápidamente) lo cual nos llevó a preguntarnos cuántas otras cosas estaríamos dejando pasar por no conocer los términos de su dialecto.

Con el objetivo de crear features orientadas por la opinión y experiencia de gente idónea, lanzamos un formulario de Google bajo el siguiente link https://forms.gle/gzxbLRsMFxMQX1fF6 con sólo las 5 preguntas que creímos pertinentes:

1. ¿Cuál es tu país de residencia? (Mexico/Otro)
2. ¿Realizas o realizaste búsquedas de propiedades por internet? (Si/No)
3. Señala cuáles consideras más importante en un anuncio (Cant Habitaciones / Cant de Baños / Metros Cuadrados / Provincia / Ciudad / Zona / Si cuenta con gimnasio, pileta o SUM / Antigüedad de la propiedad / Sus coordenadas geográficas)
4. ¿Lees las descripciones de los anuncios? (Si/No)
5. ¿Que cualidades en la descripción pueden hacerte pagar más (o menos) por una propiedad?

3.2. Difusión e Incorporaciones

La encuesta fue compartida en grupos de *Facebook* de brokers, inmobiliarias y gente interesada en comprar/vender propiedades, al momento de elaborar este informe cosechó 20 respuestas. A partir de ellas se generaron una serie de features que contabilizaban la aparición de éstos términos y sus potenciales sinónimos.

- 'Servicios'
- 'Plusvalía'
- 'Acabados'
- 'Confort'
- 'Jardín'
- 'Broker'
- 'Fotos'
- 'Alberca'

3.3. Gráficos de Resultados

Nos pareció relevante señalar las respuestas obtenidas para las preguntas [3] y [5], por ser aquellas que revisábamos frecuentemente en busca de nuevas features que no se nos hubieran ocurrido antes, empezando por los votos recibidos entre las características propuestas de las propiedades:

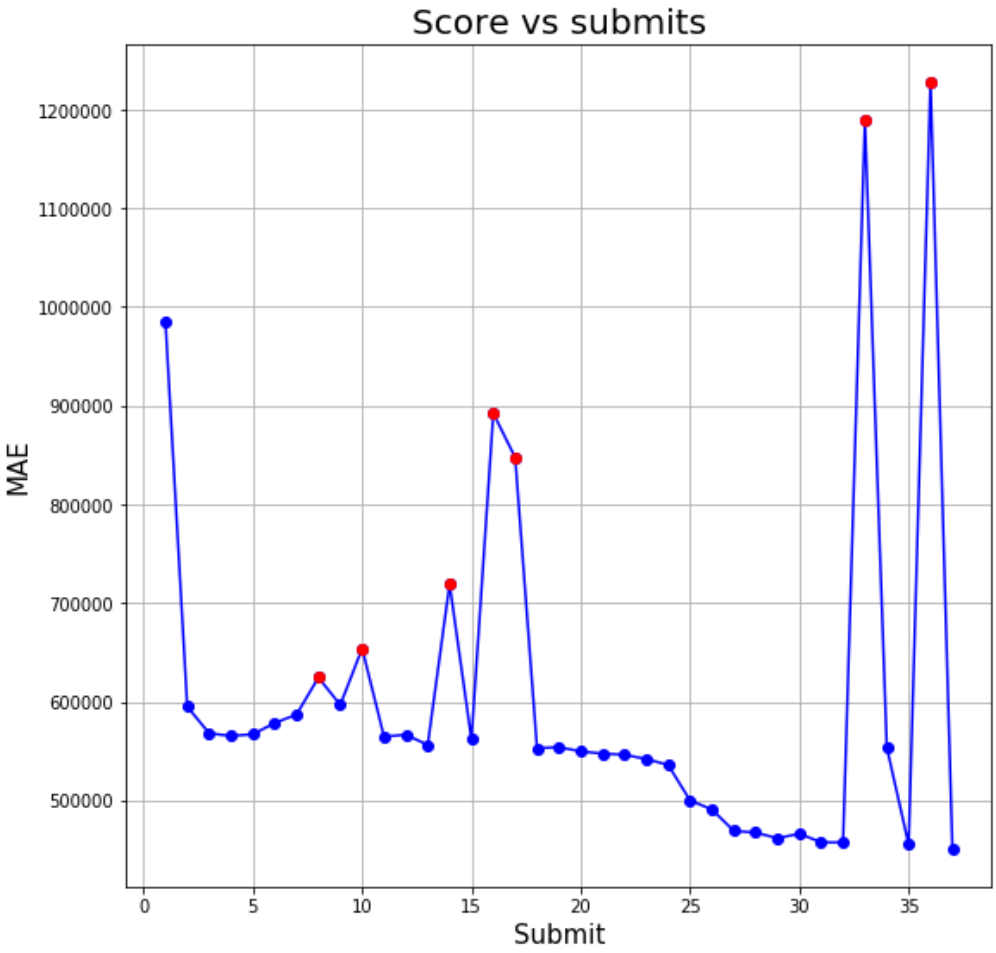
Lo que resulta muy curioso de éste gráfico es que tal como el propio XGB, las dos características que consideran más importantes los encuestados son la zona y los metros cuadrados.

A su vez, las respuestas de la pregunta [5] nos servían para hallar coincidencias entre las opiniones redactadas por los propios encuestados, a través de este *Wordcloud* pueden observarse los términos más frecuentes.

Wordcloud sobre respuestas de la pregunta [5]

4. Evolución del Score Obtenido en Kaggle

El siguiente gráfico representa el Score obtenido en en el Leaderboard privado para cada uno de los submits de las predicciones, cabe destacar que no fue confeccionado para ésta entrega sino que se le hizo un seguimiento a lo largo de toda la competencia.



Cada uno de los puntos rojos (máximos locales) fueron producidos por distintos errores en el desarrollo hasta conseguir el modelo final, entre ellos se encuentran:

- Filtración de datos por los Mean Encoding
- Tamaño insuficiente de set de test
- Submits correspondientes a distintos Modelos
- Incorporación de features sin medir su Feature Importance
- Intentos desesperados de recuperar el 1er puesto en el Leaderboard

5. Modelo Final

5.1. Features Presentes

antiguedad	habitaciones	garages
banos	metroscubiertos	metrostotales
idzona	lat	lng
fecha	gimnasio	piscina
calefaccion	suite	avenida
gim	ubicacion	balcon
camaraseg *	parque	amoblado
bañera	transporte	pileta
lujo	humilde	ventana
luz	contable *	agente
garante	finanza *	turismo
longitud_desc	longitud_titulo	cant_! *
tranquilidad	reparacion	mascotas
accesibilidad	normas	beneficios
conexion	servicios_desc	metros_desc
acabados *	plusvalia *	cocina
alberca	negacion *	variospisos
vestidor	popularidad *	aeropuerto_cercano
traslado	dueño	monoambiente
renta	oportunidad *	antiguedad_prop
niños	fotos	lavadero
$ *	trabajo	agua
amenidad *	broker	mes_publicacion
metrosdescubiertos	relacionmetros *	servicios
precioMetroMean	precioMetroCubMean	precioMetroDesMean
precioMetroMeanC	precioMetroCubMeanC	precioMetroDesMeanC
precioMeanZona	provincia_mean	ciudad_mean *
idzona_mean *	metroscubiertosminmax *	metrostotalesminmax *
metrosdescubiertosminmax *	-	-

*Marcamos algunos features que creímos necesario explicar aparte.

- camaraseg: cantidad de apariciones de palabras relacionadas con la seguridad y protección de la propiedad en título y descripción.
- acabados: cantidad de apariciones de palabras relacionadas con la calidad de los acabados o terminaciones en título y descripción.
- $: cantidad de apariciones del signo pesos "$" en título y descripción.
- amenidad: cantidad de apariciones de palabras relacionadas con las amenidades o cualidades extra en título y descripción.
- ciudad_mean: la ciudad de la propiedad con mean encoding.
- metrostotalesminmax: metros totales expresados con minmax.
- contable: cantidad de apariciones de la palabra precio en título y descripción.

- finanza: cantidad de apariciones de palabras relacionadas con créditos, préstamos, cuotas y pagos en título y descripción.
- plusvalía: cantidad de apariciones de la palabra plusvalía en título y descripción.
- negación: cantidad de apariciones de la palabra no en título y descripción.
- popularidad: cantidad de apariciones de palabras relacionadas con la fama o popularidad de la propiedad en título y descripción.
- oportunidad: cantidad de apariciones de frases que fomenten la compra de la propiedad, como por ejemplo hacer referencia a que el cliente está ante una oportunidad imperdible. Esto se midió para título y descripción de las publicaciones.
- relacionmetros: es la relación entre los metros totales sobre los metros cubiertos de cada propiedad.
- idzona_mean: es el id de la zona expresada en mean encoding.
- metrosdescubiertosminmax: es la cantidad de metros descubiertos expresada con minmax.
- cant_!: cantidad de signos de exclamación en título y descripción.
- metroscubiertosminmax: es la cantidad de metros descubiertos expresada con minmax.

5.2. Modelo Empleado

Para el modelo final lo que se realizo fue, tomar el mejor modelo de XGBoost y el mejor modelo de LightGBM y se los ensambló multiplicando las predicciones de XGBoost por 0.7 y las predicciones de LightGBM por 0.3, para finalmente sumarlas y obtener tener la mejor predicción posible.

5.3. Score Obtenido

El mejor score obtenido en la competencia fue de 451.602 que al momento de redactar éste informe nos coloca en el segundo puesto del Leaderboard publico a unos 23.000 puntos de diferencia con el primer puesto.

6. Conclusiones

- **Qué aprendimos:** Es posible realizar un análisis intenso de nuestros datos aún siendo principiantes, encontrando correlaciones entre las características de cada elemento y el target (en este caso el precio de la propiedad), como también aplicar distintos modelos y técnicas de Machine Learning, conscientes de cómo agregar o quitar complejidad a nuestro modelo puede impactar en la calidad del resultado.

- **Qué fue lo más interesante:** Para este set de datos en particular, lo más interesante fue hallar los elementos más determinantes del precio de una propiedad (la zona en la que está ubicada y el costo promedio para ella de los metros cuadrados, por amplia diferencia frente a todas las otras) y que las respuestas de la encuesta nos ayuden a identificar buenas features. En cuanto a técnicas de Machine Learning, algo que nos llamó mucho la atención fue ver que un modelo que fitea muy rápidamente (como LightGBM) puede obtener predicciones casi igual de buenas que uno que lleva horas (como XGBoost).

- **Qué nos quedo pendiente:** Nos quedó pendiente implementar un modelo viable de blending para el ensamble de datos, la construcción de una red neuronal más diseñada acorde a lo necesitado, y en última instancia, también nos quedamos con ganas de recuperar el primer puesto en la competencia.