


7506 - Organización de Datos

Trabajo Práctico N° 2: Machine Learning

Auspiciado por: Navent

Alumnos:

Cozza, Fabrizio	fabrizio.cozza@gmail.com	97.402
Rozanec, Matías	rozanecm@gmail.com	97.404

 https://github.com/rozanecm/7506_tp2_2c2019

2c2019

Facultad de Ingeniería - Universidad de Buenos Aires

Índice

Índice	1
Introducción	2
Objetivos	2
Herramientas utilizadas	2
Locales	2
Cloud	3
Acerca del problema	3
Feature selection	4
Correlation matrix	4
Feature importance	5
BoostARoota	5
Feature engineering (134 features probados en total y algunas modificaciones sobre features)	6
Variable objetivo (2 features)	6
Asimetría estadística	6
Imputación de valores faltantes	6
Entrenamiento de modelos para predecir valores faltantes	7
Datos faltantes en LightGBM	7
Features categóricos (47 features)	7
Fecha (15 features)	9
Amenities (20 features)	9
Features de texto (47 features)	9
Metros cuadrados (1 feature)	11
Clustering (2 features)	12
Algoritmos utilizados (15)	14
Metodologías para mejorar los algoritmos	14
Búsqueda y tuning de hiper-parámetros	14
Líneas de desarrollo pendientes	15
Conclusiones	15
Anexo	16
Gráfico Score vs Método	16

Introducción

En este informe se presentan los razonamientos y propuestas para la realización del segundo trabajo práctico de la materia.

En el mismo se presentan primero las herramientas utilizadas, y luego se procede a desarrollar los distintos enfoques que se fueron probando durante el transcurso del trabajo práctico en las distintas secciones del mismo (feature selection, feature engineering, modelos de entrenamiento), detallando los descubrimientos y aprendizajes que se fueron dando en cada etapa.

Objetivos

El objetivo del trabajo práctico es poder aprender, entrenar y aplicar los conocimientos adquiridos durante la segunda parte de la materia utilizando para esto un caso real, con datos reales, con el fin de obtener los mejores resultados posibles en la competencia propuesta en la plataforma Kaggle.

Lo único que distó de la realidad fue la dimensionalidad de los datos y gracias a esto no solo no se alteraron los aprendizajes sustraídos de la experiencia, sino que también se hizo posible explorar una amplia gama de posibilidades que de contar con datos mucho más pesados hubiese resultado imposible.

Herramientas utilizadas

Locales

El entorno de trabajo utilizado fue Jupyter-lab. Se hizo uso extensivo de la extensión toc¹, sin la cual a medida que el trabajo empezó a crecer en tamaño, se hubiese tornado inmanejable. Esto permitió también reducir la cantidad de notebooks, creando notebooks nuevos solamente en casos de probar cosas enfoques radicalmente distintos o auxiliares al trabajo principal. De esta forma, hay un solo notebook² en que se probó la mayoría de los modelos, pero como el mismo está bien segmentado, ordenado y con fácil navegación mediante el índice, finalmente resultó muy entendible a pesar de la extensión que fue adquiriendo.

La principal herramienta utilizada fue la librería scikit-learn, que proporciona la mayoría de las primitivas necesarias tanto para la transformación de datos como para el entrenamiento de modelos, potenciando esto último con herramientas de búsqueda de hiper parámetros,

¹ toc: table of contents. <https://github.com/jupyterlab/jupyterlab-toc>

² Para evitar colisiones de trabajo de difícil resolución en git, cada alumno trabajó en notebooks separados, por lo que cuando se habla 'del notebook con modelos', realmente se refiere a mínimo uno por alumno para prueba de diferentes enfoques.

ampliando las posibilidades de trabajo con herramientas específicas que evitan que se filtre información entre los sets de test y entrenamiento mediante pipelines.

Se destaca el papel de los mencionados pipelines, porque además de la ventaja aludida en el párrafo anterior, la naturaleza compacta de los mismos permitió trabajar de forma ordenada las múltiples transformaciones que se fueron probando sobre los diversos grupos de features.

Aparte de scikit, se utilizaron librerías las librerías LightGBM, XGBoost, Catboost, Keras, GloVe, spacy, BoostARoota³ y stop-words⁴.

Cloud

Con la gran cantidad de cosas que se empezó a ver se podrían ir probando, pero más que nada con la necesidad evidente de buscar hiper parámetros para los algoritmos que se quiso probar, surgió rápidamente el deseo de poder aumentar el poder de cómputo de alguna forma. Una gran idea ha sido subir los datos necesario a Kaggle (de forma privada, con acceso restringido únicamente para los miembros del grupo), pudiendo aprovechar así las máquinas virtuales ofrecidas por ellos, en las que se dispone de 9 horas donde se pueden correr hasta 5 notebooks en simultáneo.

Hay varios modelos que en los notebooks locales solamente fueron planteados, pero donde no se ven los resultados finales porque los mismos se obtuvieron desde notebooks en Kaggle.

Hay que notar de todas formas que esto tiene también sus límites: una de las últimas cosas que se intentaron fue entrenar un modelo para el cual localmente 10 tasks fueron completadas en 2.7 segundos mientras que en Kaggle esas mismas 10 tasks consumieron sus 57 segundos.⁵ Probablemente se esté usando hardware compartido en Kaggle, y dependa un poco de la carga que tienen los servidores en el momento, pero la diferencia fue en este caso demasiado marcada, por lo que finalmente hay sentimientos mixtos con esta opción: brinda excelentes posibilidades de paralelizar mucho el trabajo, pero hay que tener cuidado con estas cosas, porque es probable que algo no esté andando según lo esperado.

Acerca del problema

El problema planteado refiere a la predicción del precio de propiedades inmobiliarias en México, principalmente de la zona céntrica del país.

Al tener la posibilidad de ver cómo se distribuye la variable del precio en el primer trabajo práctico y ver cómo se relaciona con las demás características de las propiedades,

³ <https://github.com/chasedehan/BoostARoota>

⁴ <https://github.com/Alir3z4/python-stop-words>

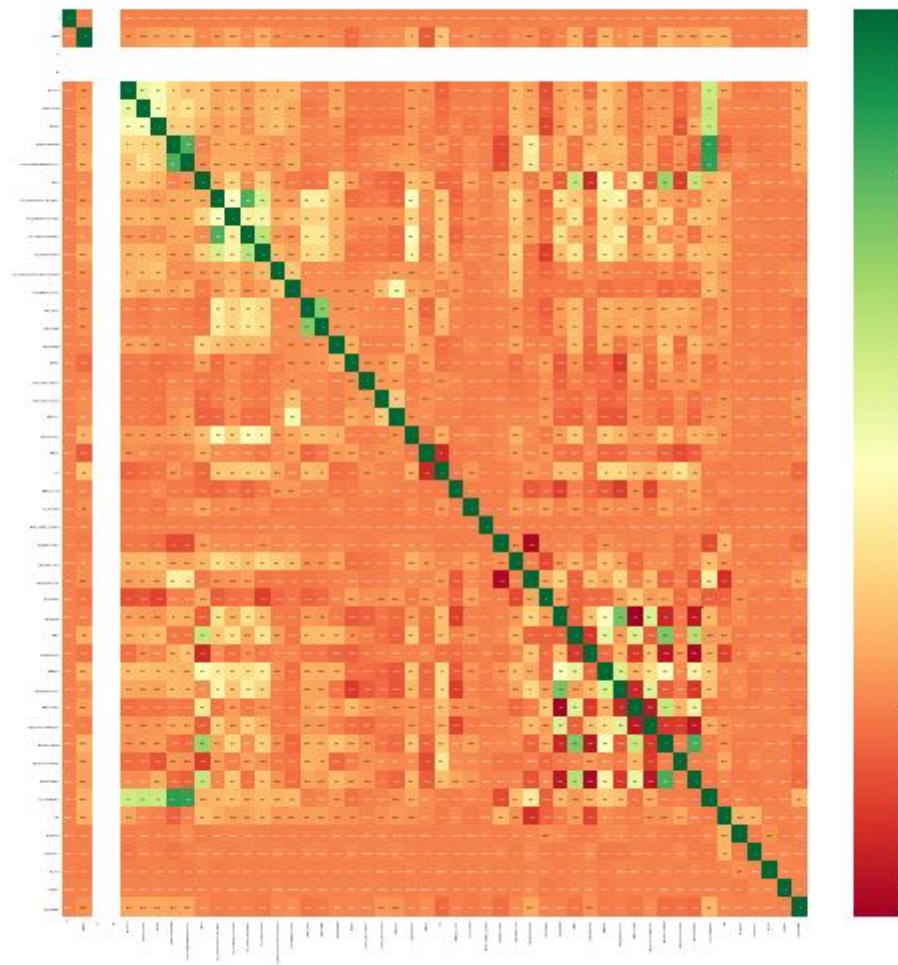
⁵ Estas mediciones pueden parecer rápidas, pero al principio siempre se hizo una corrida de prueba con pocos registros para asegurarse de que el código corre bien, y una vez chequeado que todo está en orden, se procedía a correr el código con todos los datos.

podemos afirmar que se podría usar y que nos basamos para la resolución del problema en el enfoque de algoritmos de regresión.

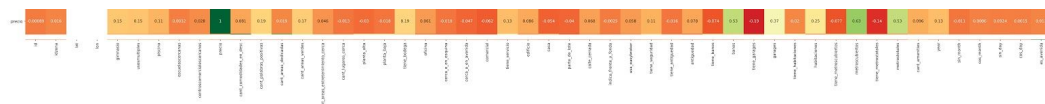
Feature selection

Correlation matrix

Un enfoque básico era ver la matriz de correlación, en especial ver la correlación entre el target y todas las columnas y dejar aquellas que tenían mayor correlación con la misma. El enfoque no dio resultados positivos y dió una idea inicial de que remover features no era beneficioso.



Se recortó a continuación la fila del target y su relación con las demás variables:



Se dejaron para la prueba mencionada que dio resultados negativos aquellas con valores mayores a 0.09 siendo estas: gimnasio, usos multiples, piscina, cant palabras positivas, cant areas verdes, tiene bodega, tiene servicio, tiene seguridad, banos, garages, habitaciones, metros cubiertos, metros totales, year, cant amenities.

Para ver la imagen en mayor detalle por favor ir al [github](#).

Feature importance

Una forma de probar si features nuevos aportaban algo se basó en entrenar algún algoritmo (random forest o LightGBM), y estudiar sus feature importances. Se eligieron los dos algoritmos mencionados debido a que LightGBM fue el algoritmo que venía dando mejores resultados, y Random Forest porque es que da una importancia que puede ser usada luego en otros modelos también.

Si después de sacar los features con peor puntaje al calcular la feature importance el algoritmo obtenía mejores resultados, se decidió descartar dichos features.

BoostARoota

Este algoritmo es básicamente un XGBoost más eficiente y enfocado a la selección de features.

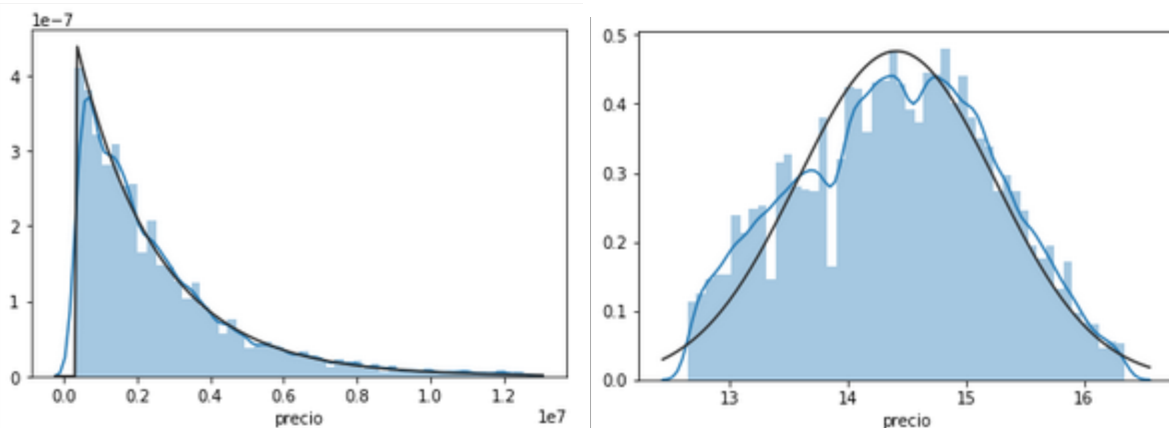
Siendo este algoritmo ya un método más confiable, lo que hace es devolver cuántos features son importantes dentro del set de datos. Raramente el algoritmo mostraba que servían todos los features.

Con cierto parámetro se hacía al algoritmo más restrictivo pero había que bajarlo mucho para que haya una reducción de features. Una vez que se intentó este cambio y se vió el posible uso de una menor cantidad de features, se testeó y el resultado no mejoró así que fue en general fué mejor utilizar todos los features posibles durante el trabajo práctico y enfocarse en la creación de nuevos features.

Feature engineering (134 features probados en total y algunas modificaciones sobre features)

Variable objetivo (2 features)

En primer lugar cabe destacar que a partir de la observación hecha en el análisis exploratorio de datos de la distribución de la variable objetivo se probó el comportamiento de los algoritmos antes y después de tomarle logaritmo, y se pudo comprobar que efectivamente el resultado mejora notablemente.



Otra cosa que se tomó en cuenta fue la inflación del peso mexicano a lo largo del tiempo: desde enero de 2012 hasta diciembre de 2016 hubo un 17.48% de inflación. Si bien estamos acostumbrados a números mucho mayores, al haber muchos registros de propiedades a predecir, esto puede ir induciendo errores fácilmente evitables que acumulados por la cantidad de registros terminen teniendo un peso mucho mayor al esperado intuitivamente. Para solucionar esto, se tomó la cotización del dólar por día, y una vez predecido el precio en dólares, se volvió a pasar a pesos mexicanos. Sorprendentemente, esto no indujo ninguna mejora en los resultados.

Asimetría estadística

Para los features numéricos se calculó el valor de la asimetría estadística (skewness), y se aplicó una transformación logarítmica a las variables cuyo valor superará 0.5.

Contrario a lo que sucedió con la variable objetivo, este cambio no indujo mejoras en los resultados.

Imputación de valores faltantes

En el caso de los valores faltantes, se tomaron siguientes caminos:

- En casos de datos numéricos o booleanos, imputar con el valor más frecuente

- En casos de features categóricos, se imputó con un string vacío
- Entrenamiento de modelos para predecir valores faltantes
- En el caso de LightGBM, no se completaron

Entrenamiento de modelos para predecir valores faltantes

Por cada variable con valores faltantes, se creó un feature indicando si el feature es faltante o no, de forma que cuando queden rellenos todos los datos, todavía se pueda distinguir cuáles estuvieron originalmente presentes y cuáles no.

A continuación se prepararon los datasets para la predicción: tanto de train como del test set originales, se juntaron los registros sin datos faltantes en un nuevo train set, y se juntaron los registros restantes en el set a predecir.

En todos los casos se usó LightGBM con grid search.

Ante la tarea de tener que predecir features que son numéricos, pero que a la vez cuentan con un número limitado de respuestas posibles, surgió la duda: habría que usar clasificación o regresión? Basados en una respuesta a esta pregunta en Cross Validated (Stack Exchange)⁶, se decidió finalmente usar regresión, porque no importaba el número en sí, sino mas bien una representación numérica que represente lo mejor posible ese feature en el contexto de una predicción intermedia, o sea, que el feature será luego usado para predecir otras cosas todavía. Así, es preferible que se prediga que una propiedad tendría 3.5 baños a estrictamente 3 o 4 por ejemplo.

De todas formas, cada uno de los features con valores faltantes podría ser un problema en sí mismo, por lo que está claro que se podría indagar muchísimo en cada caso particular. Como fue usado como paso intermedio, no se indagó mucho más en esto.

Datos faltantes en LightGBM

Internamente, LightGBM (y XGBoost) ignoran los valores faltantes en el cálculo del split, y luego los deja del lado que minimiza el loss.⁷

Features categóricos (47 features)

Los features categóricos fueron divididos en dos categorías en base a la cantidad de valores posibles que podían tomar: grandes y chicos. En ambos casos lo se los procesó con one hot encoding seguido de una reducción de dimensiones con SVD.

Las variables tipo de propiedad y provincia son las que toman una cantidad menor de valores posibles, mientras que la variable ciudad tiene un espacio de direcciones mucho mayor.

⁶

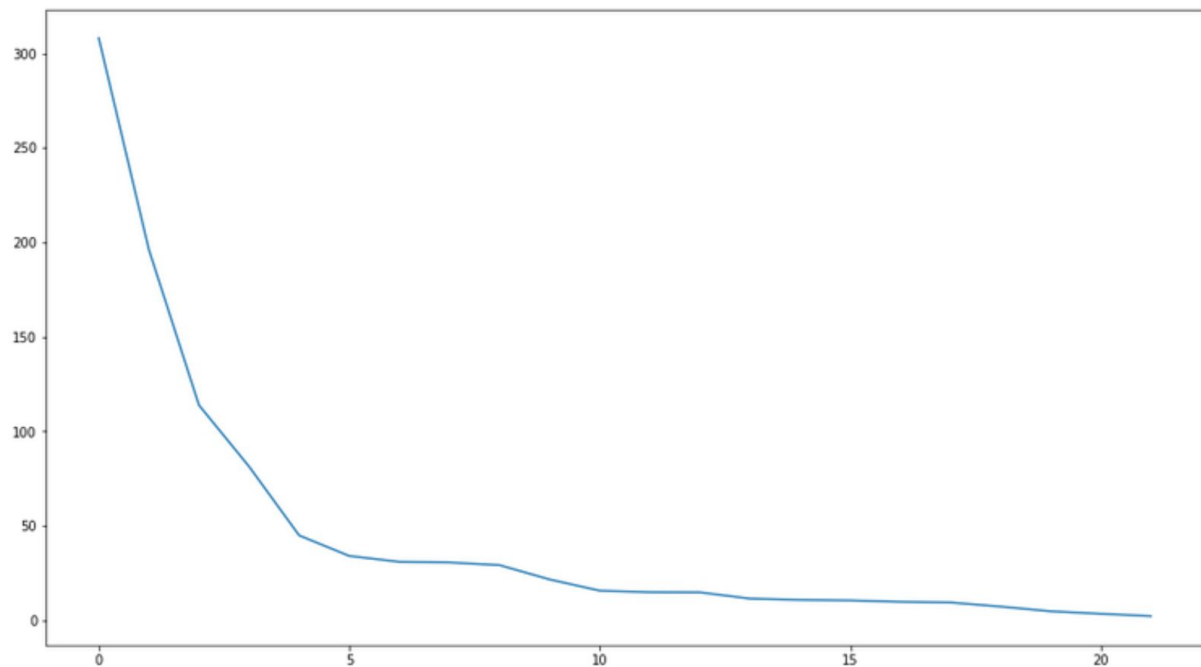
<https://stats.stackexchange.com/questions/282803/response-is-an-integer-should-i-use-classification-or-regression>

⁷ <http://mlexplained.com/2018/01/05/lightgbm-and-xgboost-explained/>

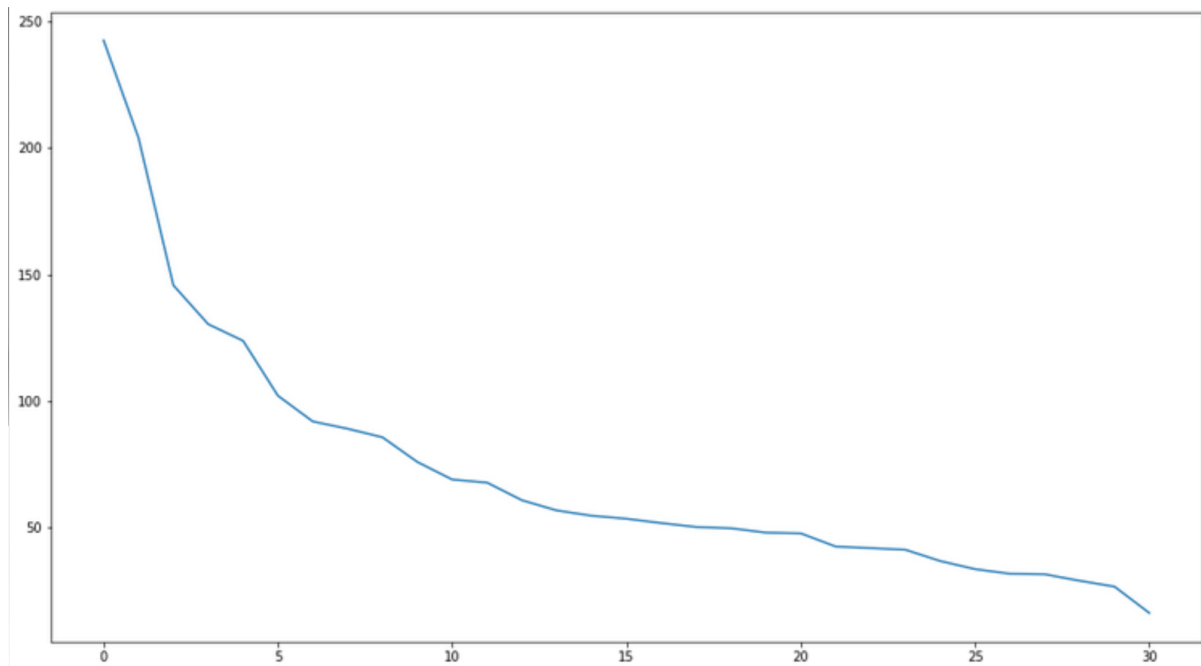
La razón que justifica esta división es que en los dos casos de las variables que quedaron con el mismo procesamiento, la reducción de dimensiones se reduciría a la misma cantidad de dimensiones.

La elección de la cantidad de dimensiones se hizo mediante el método del codo, considerando la energía de la matriz.

Método del codo tipo de propiedad:



Método del codo provincia:



Por otro lado, también se hizo mean encoding de los features categóricos.

Fecha (15 features)

La variable fecha en su formato entero no puede alimentar ningún algoritmo, por lo que se decidió:

- extraer el año, mes y día como features aparte
- los features mes y día se desdoblaron en otros dos features que consisten en tomarle el seno y el coseno a cada feature. De esta forma se llega a contemplar la naturaleza cíclica de estos features.

Además de los features mencionados, se probó agregar el día del año, cuatrimestre al que pertenece la fecha, indicador de año bisiesto, cantidad de días que tiene el mes, si pertenece al inicio/fin de mes o inicio/fin de año y semana del año, pero ninguno de ellos aportó mejoras a los algoritmos.

Amenities (20 features)

Un feature que contribuyó mejoras fue la de contar la cantidad de amenities por propiedad.

Por otro lado, se crearon nuevas variables booleanas mediante operaciones AND, OR y XOR entre cada par de amenities, pero lo mismo no sólo no mejoró el puntaje, sino que lo empeoró notablemente. Aparentemente, todo esto fue percibido como ruido.

La intuición detrás de este enfoque radica en explicitar las relaciones entre amenities.

Features de texto (47 features)

Sobre feature **dirección**:

- Nuevo feature indicando si la propiedad se ubica sobre una avenida o no.

Sobre feature **título**:

- Nuevo feature indicando la cantidad de tags html que contiene
- Nuevo feature indicando la cantidad de palabras
- Nuevo feature indicando la cantidad de palabras únicas
- Nuevo feature indicando diversity score, calculado como el ratio entre cantidad de palabras únicas y cantidad de palabras
- Nuevo feature indicando la cantidad de caracteres
- Nuevo feature indicando la cantidad de signos de puntuación
- Nuevo feature indicando la entropía de Shannon
- Nuevo feature indicando la longitud promedio de las palabras

Sobre feature **descripción**:

- Basado en los word clouds del TP1:
 - Nuevo feature indicando la cantidad de comodidades

- Nuevo feature indicando la cantidad de palabras positivas (para dar ejemplo del criterio: “bonita”, “hermosa”, “excelente”, etc...)
- Nuevo feature indicando la cantidad de áreas dedicadas a cierta actividad (para dar ejemplo: “sala comedor”, “cuarto lavado”, etc...)
- Nuevo feature indicando la cantidad de áreas verdes
- Nuevo feature indicando la cantidad de areas de entretenimiento
- Nuevo feature indicando la cantidad de lugares cerca ya sean restaurantes, el metro, etc...
- Nuevo feature indicando si tiene planta alta o no
- Nuevo feature indicando si tiene planta baja o no
- Nuevo feature indicando si tiene bodega o no
- Nuevo feature indicando si tiene oficina o no
- Nuevo feature indicando si está en una esquina o si está cerca de una esquina
- Nuevo feature indicando si está en una avenida o si está cerca de una avenida
- Nuevo feature indicando si es area comercial
- Nuevo feature indicando si tiene algun tipo de servicio
- Nuevo feature indicando si es un edificio
- Nuevo feature indicando si es una casa
- Nuevo feature indicando si es parte de un lote
- Nuevo feature indicando si está en una calle cerrada
- Nuevo feature indicando si muestra las medidas del frente y del fondo
- Nuevo feature indicando si usa easybroker
- Nuevo feature indicando si tiene algun tipo de seguridad
- Basado en la investigación de features de texto:
 - Nuevo feature indicando la cantidad de stopwords
 - Nuevo feature indicando la cantidad de signos de puntuación
 - Nuevo feature indicando la cantidad de palabras
 - Nuevo feature indicando la cantidad de caracteres
 - Nuevo feature indicando la ocurrencia del top10 de trigramas
 - Nuevo feature indicando la ocurrencia del top10 de bigramas
 - Nuevo feature indicando la cantidad de numeros
 - Nuevo feature indicando la longitud promedio de las palabras
 - Nuevo feature indicando diversity score, calculado como el ratio entre la cantidad de caracteres y cantidad de palabras
 - Nuevo feature indicando la cantidad de las palabras mas usadas
 - Nuevo feature indicando la cantidad de las palabras menos usadas
 - Nuevo feature indicando la cantidad de palabras con las que suelen terminar con cierto caracter
 - Nuevo feature indicando la cantidad de palabras con las que suelen empezar con cierto caracter
 - Nuevo feature indicando el sentimiento del texto
 - Nuevo feature indicando la entropia de Shannon
 - Nuevo feature indicando la cantidad de tags html que contiene
 - Nuevo feature indicando la cantidad de palabras únicas

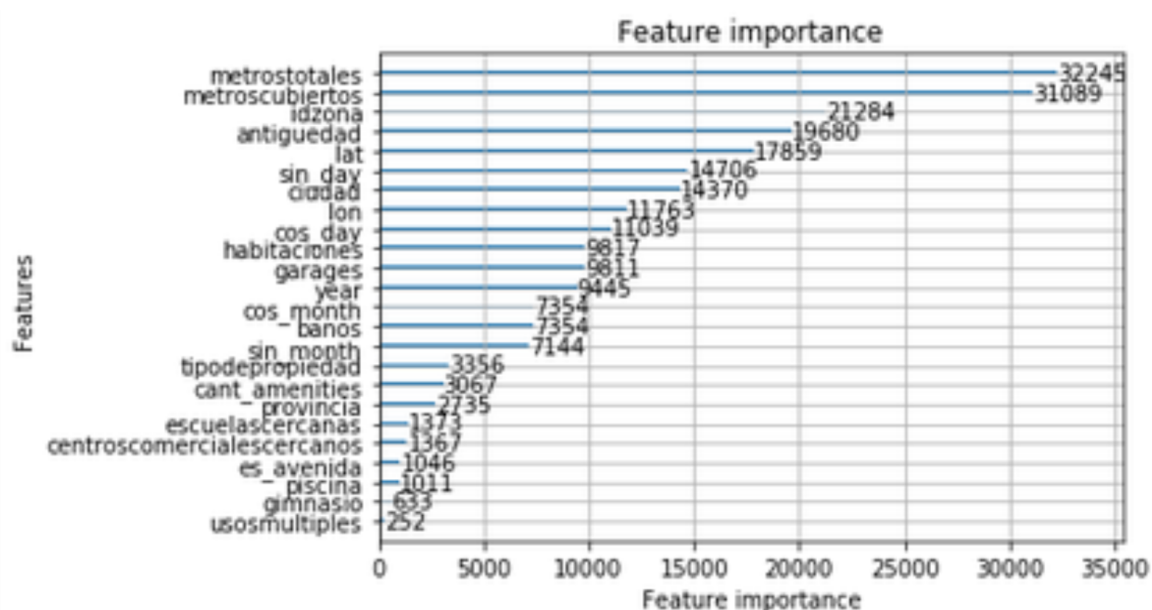
En cuanto al comportamiento de las features en líneas generales estas features siempre se agregaron de a grupos y dieron una mejora al puntaje, lo que tiene sentido porque desglosan las features de texto de alguna manera.

Al ser agregadas de manera grupal y no progresivamente, una mejora que se podría hacer y quedará pendiente es la de probar una por una para localizar aquellas que realmente mejoran el puntaje y dejar de lado aquellas que no.

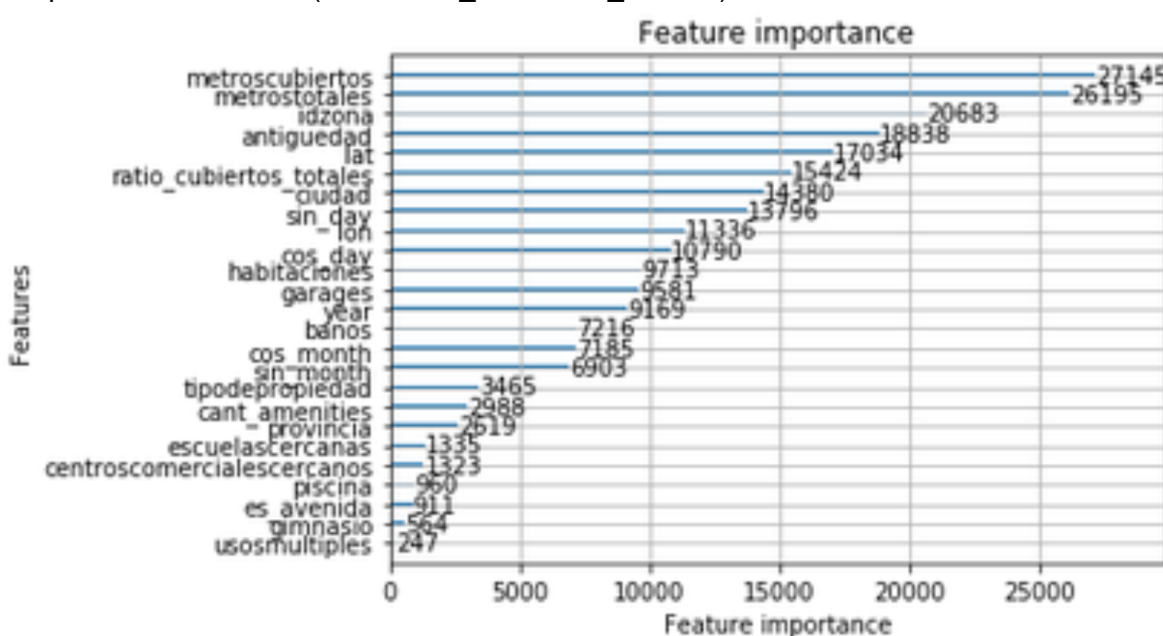
Metros cuadrados (1 feature)

Se creó un feature indicando el ratio entre los metros cubiertos y los totales.

Antes de la creación:



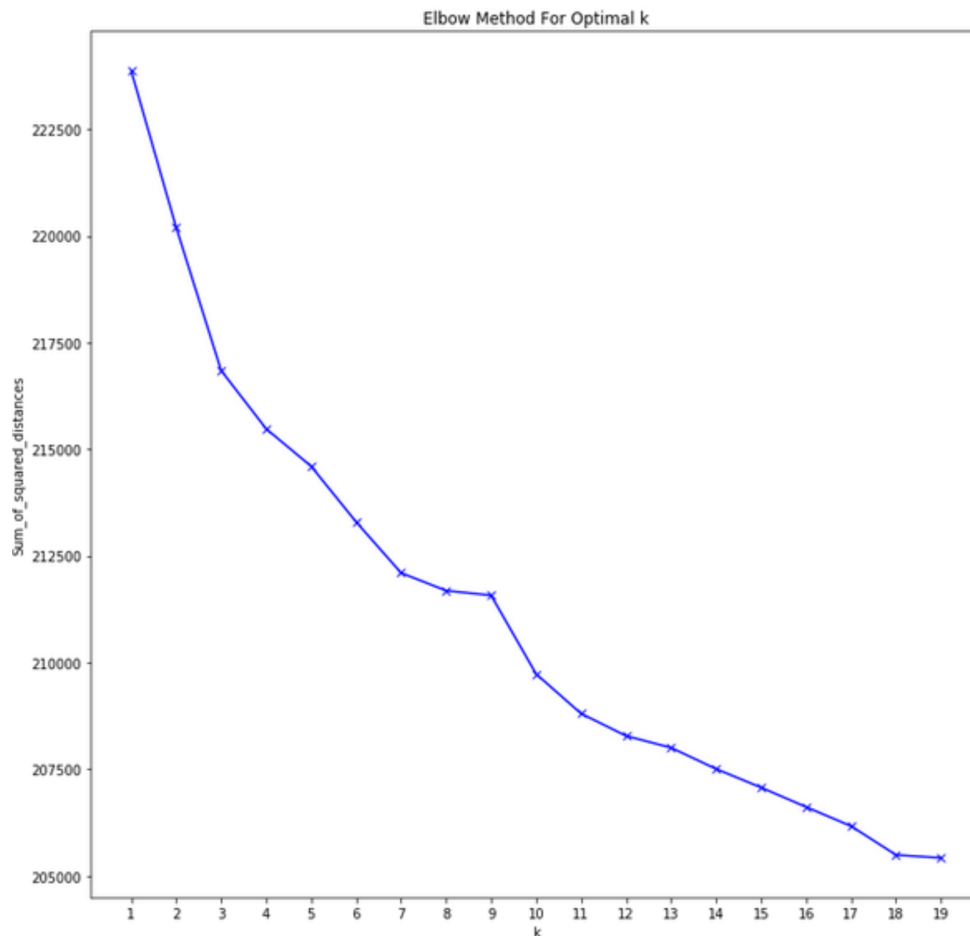
Después de la creación (notar **ratio_cubiertos_totales**):



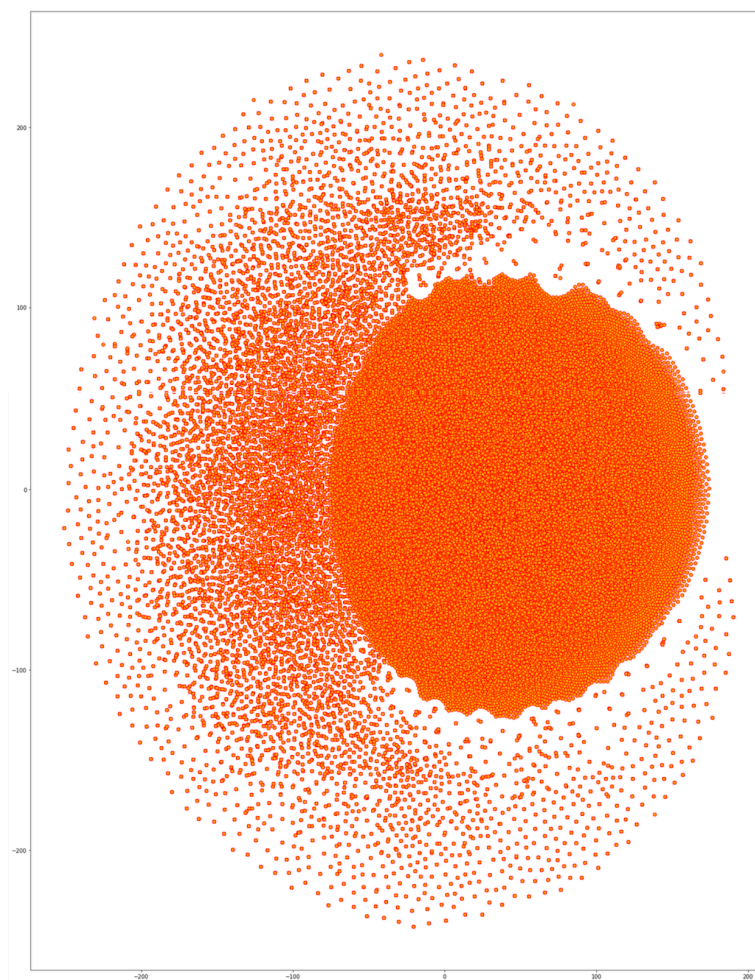
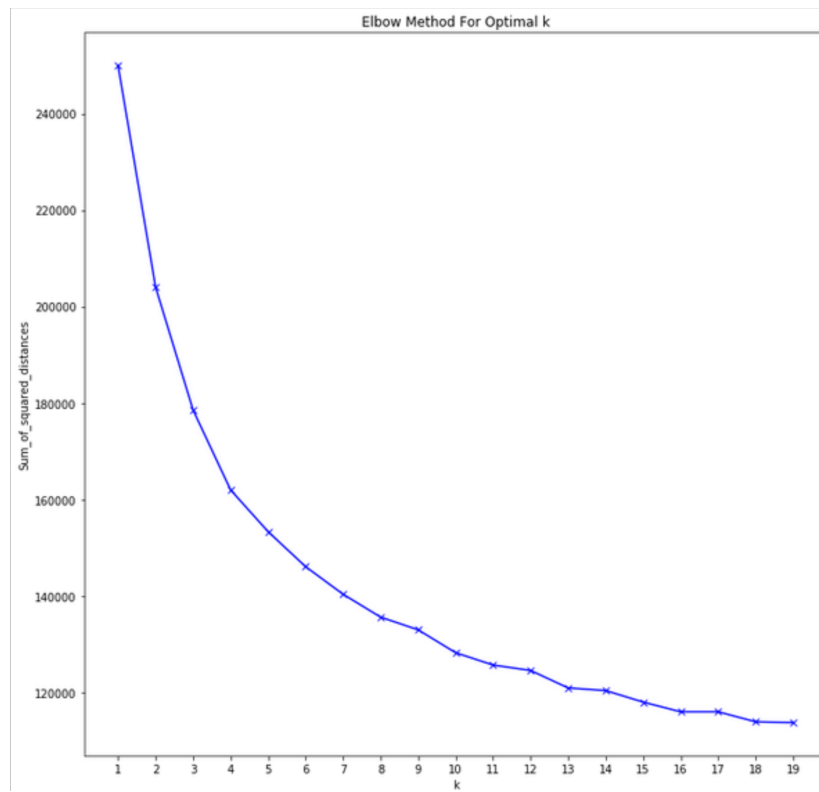
Clustering (2 features)

Se utilizó clustering por medio de K-Means para el feature de descripción con dos enfoques:

- *Primer enfoque:* Se calculó el puntaje de TF-IDF para cada campo de descripción y a este vector se le midió con Min Batch K Means (para no usar toda la dimensión de los datos) con el método del codo la cantidad de dimensiones necesitadas para clustering dando como resultado 9 clusters. Luego sí se utilizaron todos los datos para generar un feature con K Means indicando a qué cluster correspondía cada campo.



- *Segundo enfoque:* Se generó un vector con embedding por medio del llamado GloVe method para cada campo de descripción y al igual que el método anterior con el método del codo se determinó la cantidad de dimensiones necesitadas para clustering dando como resultado 9 clusters. Luego se utilizaron todos los datos para generar un feature con K Means indicando a qué cluster correspondía cada campo.



Relación entre puntos con TSNE

La utilización de estos features empeoró drásticamente cualquier método al que se le probara. Esto tiene consistencia con los gráficos de los métodos de codo ya que hay más una relación lineal que un verdadero codo como también se puede ver en el gráfico realizado con TSNE que muestra la relación entre los puntos. Entonces si bien se probó no se recomienda utilizarlo.

Algoritmos utilizados (15)

A continuación se listan los algoritmos probados durante la realización del trabajo práctico:

- Stacking ensemble ★
- XGBoost
- Catboost
- LightGBM 🏰
- Neural network with Keras
- Random forests regressor
- Promedio de los mejores enfoques
- Linear SVR (SVM)
- Hist Gradient Boosting regressor
- Linear regression
- SGD regressor
- Bagging regressor
- Extra Trees regressor
- Gradient boosting
- KNN

★: El algoritmo que mejor score dió

🏰: El algoritmo base que mejor funcionó y más se utilizó

Metodologías para mejorar los algoritmos

Para cualquier tipo de prueba sobre los mencionados algoritmos siempre se utilizó el método train test split y las métrica de MAE (mean absolute error) o RMSE (root mean squared error) para estimar la correcta funcionalidad del algoritmo.

Búsqueda y tuning de hiper-parámetros

A todos los algoritmos usados se les realizó una búsqueda de hiper parámetros mediante grid search cross validation y en alguno en particular se probó random search cross validation.

La razón por la que se prefirió usar grid search en la mayoría de los casos radica en que si bien la cantidad de hiper parámetros solía ser muy grande, se prefirió tener más control sobre las alternativas que se prueban.

Líneas de desarrollo pendientes

Una de las posibles cosas a expandir en el trabajo es profundizar el análisis de texto mediante word embedding. En un artículo en Towards Data Science⁸ se mencionan 3 posibles métodos para lograr, de los cuales solamente se pudo probar uno.

Además los features de texto tuvieron modelos dedicados en una fase muy tardía, porque se trabajó más con el resto de los features. Esto deja mucho margen para seguir trabajando todavía. Probablemente sería bueno ahondar en modelos de texto basados en redes neuronales.

Por último, si bien la cantidad de features agregados no es menor, probablemente se pueda trabajar todavía mucho más sobre el asunto. Metodologías repetitivas como la búsqueda de hiper-parámetros pueden llevar a los algoritmos a mejorar hasta cierto punto pero la creación de features es un área mayor a explorar y no tan limitada.

Conclusiones

Habiendo finalizado el trabajo práctico, se puede decir que se ha logrado un entendimiento bastante sólido de muchos de los temas vistos en clase. Se pudieron aplicar muchas técnicas de trabajo de forma organizada logrando mucha comodidad en el manejo de las librerías y los conceptos que se suelen manejar en el ámbito.

El formato del tp también resultó muy bien, porque al estar contrastando todo el tiempo el score en el Leaderboard, uno tiene un feedback constante de cómo viene haciendo las cosas, y esto fue algo que siempre mantuvo al grupo con cierta consciencia de cómo se venía trabajando. Si bien el grupo llegó a estar en el primer puesto, habiendo finalizado en el noveno lugar (al momento de escribir este informe) es un resultado más que aceptable.

Se destaca como muy positivo el hecho de haber contado con un set de datos variado pero de tamaño reducido que permitió probar muchísimas cosas con bastante facilidad.

Por último, además de haber aplicado el conocimiento obtenido en clase, el problema concreto fue una motivación extra muy grande para salir a leer y aplicar cosas variadas de estos temas.

La recomendación para aquel que tenga que trabajar con estos datos en un futuro es la de por un lado intentar diferentes métodos de encoding para las variables categóricas ya que son importantes, como también intentar extraer la mayor cantidad de información sobre los campos de texto. En un futuro quizá averiguar y aprender bien de qué va lo nuevo en extracción de features en texto como por ejemplo embedding, podría llevar al texto a una utilidad mayor.

⁸ <https://towardsdatascience.com/what-the-heck-is-word-embedding-b30f67f01c81>

Anexo

Gráfico Score vs Método

