FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

[75.06 / 95.58] Organización de Datos
Primer cuatrimestre de 2021

Grupo: Equipo onda maravilla lobo

Alumno	Padrón	Mail
Dituro, Celeste	104011	cdituro@fi.uba.ar
Pfaab, Ivan	103862	ipfaab@fi.uba.ar
Lopez, Victoria Abril	103927	vlopez@fi.uba.ar

Repositorio: https://github.com/vickyylopezz/Organizacion-de-Datos-75.06-95.58-/tree/main/TP2
Video: https://youtu.be/fhzUsk5bA1M

Curso

- Argerich, Luis Argerich
- Golmar, Natalia
- Martinelli, Damian Ariel
- Ramos Mejia, Martín
- Gianmarco Cafferata
- Joaquin Torre Zaffaroni
- Julian Crespo
- Esteban Djeordjian

Índice

1. Introducción

Para este trabajo practico el objetivo es resolver un problema de machine learning con los datos del TP1. Puntualmente, se debe predecir la variable 'damage_grade' que representa el nivel de daño recibido por la edificación en el terremoto que tuvo lugar en Nepal en el año 2015. El error de la solución se calculará con la métrica F1 Score, particularmente con la variación no binaria micro averaged F1 score.

$$F_{micro} = \frac{2 \cdot P_{micro} \cdot R_{micro}}{P_{micro} + R_{micro}}$$

$$P_{micro} = \frac{\sum_{k=1}^{3} TP_k}{\sum_{k=1}^{3}(TP_k + FP_k)}, \quad R_{micro} = \frac{\sum_{k=1}^{3} TP_k}{\sum_{k=1}^{3}(TP_k + FN_k)}$$

2. Feature Engineering

2.1. Limpieza de datos

Para comenzar con el proceso de Feature Engineering contemplamos no tener edificaciones repetidas y evaluar que hacer con los datos nulos. Esta limpieza se llevo a cabo en el TP1, llegando a la conclusión que no hay ni datos nulos ni edificios repetidos.

2.2. Análisis de features

Identificamos el tipo de dato de los features. De acuerdo al modelo estos pudieron verse modificados mediante una codificación o no. Para los casos numéricos tuvimos en cuenta si existe una relación de orden entre los valores para comprender mejor el comportamiento de los modelos frente a ellos.

2.3. Creación de features

Observamos cuáles eran los features más importantes y los combinamos para así crear nuevas variables.

A nuestro parecer las caracteristicas fisicas de las edificiaciones podrian ser los principales causantes del grado de daño. Como primera instancia propusimos las siguientes características:

2.3.1. Volúmen

Multipliación entre los features *'area_percentage'* y *'height_percentage'*. Este feature tiene tipo de dato entero y una relación de orden.

2.3.2. Combinación de materiales

Suma de los features *'has_superstructure_adobe_mud', 'has_superstructure_mud_mortar_stone', 'has_superstructure_stone_flag', 'has_superstructure_cement_mortar_stone', 'has_superstructure_mud_mortar_br*

'has_superstructure_cement_mortar_brick', 'has_superstructure_timber', 'has_superstructure_bamboo', 'has_superstructure_rc_non_engineered', 'has_superstructure_rc_engineered', 'has_superstructure_other. Este feature tiene tipo de dato entero y una relación de orden.

2.3.3. Combinación de usos

Suma de los features *'has_secondary_use_agriculture', 'has_secondary_use_hotel', 'has_secondary_use_rental', 'has_secondary_use_institution', 'has_secondary_use_school', 'has_secondary_use_industry', 'has_secondary_use_health_post', 'has_secondary_use_gov_office', 'has_secondary_use_use_police', 'has_secondary_use_other'*. Este feature tiene tipo de dato entero y una relación de orden.

2.3.4. Relación área y altura

División entre los features *'area_percentage'* y *'height_percentage'*. Este feature tiene tipo de dato entero y una relación de orden.

2.3.5. Antigüedad de la estructura piedra-barro

Multiplicación de los features *'age'* y *'has_superstructure_mud_mortar_stone'*. Este feature solo afectará a las edificaciones que poseen la estructura barro-piedra. Tiene tipo de dato entero y una relación de orden. Tuvimos en cuenta unicamente este tipo de estructura por el análisis realizado en el Trabajo Práctico 1, donde llegamos a la conclusión que los edificios construidos a partir de este tipo de estructura recibieron mayor impacto.

2.3.6. Promedio de la estructura barro-piedra por zona

Promedio de las edificaciones con la estructura barro-piedra por zona. Utilizamos los features *'has_superstructure_mud_mortar_stone'* y *'geo_level_1_id', 'geo_level_2_id', 'geo_level_3_id'*. Es una feature numérica con una relación de orden.

2.3.7. Media de edad por zona

Media de la edad de las edificaciones por zona. Utilizamos los features *'age'* y *'geo_level_1_id', 'geo_level_2_id', 'geo_level_3_id'*. Es una feature numérica con una relación de orden.

2.3.8. Media de volumen por zona

Media del volumen de las edificaciones por zona. Utilizamos los features *'volumen'* y *'geo_level_1_id', 'geo_level_2_id', 'geo_level_3_id'*. Es una feature numérica con una relación de orden.

2.4. Encoding

Dado que la mayoria de los modelos de Machine Learning no soportan variables categoricas, decidimos codificar las mismas a partir de One Hot Encoding. Este método agrega una columna binaria por cada valor posible que puede tomar el feature.

3. Modelos

Utilizamos cuatro modelos, uno de bagging y tres de boosting. A la hora de probar los distintos modelos de Machine Learning, hicimos una corrida inicial con los hiperparámetros por defecto para tener una predicción inicial. Luego fuimos realizando distintas búsquedas de hiperparámetros para llegar al mejor resultado. Nos centramos en los algoritmos de boosting ya que notamos mejores resultados.

3.1. Búsqueda de hiperparámetros

Necesitamos encontrar el conjunto óptimo de hiperparámetros para luego encontrar el modelo que mejor prediga. En algunos modelos este proceso es más crítico que en otros.

3.1.1. Grid Search

Establecemos un conjunto de valores a probar por cada hiperparámetro. Probamos todas las combinaciones posibles y nos quedamos con la mejor. Una vez que encontramos los valores optimos podemos refinar la busqueda ajustando los rangos. El principal problema es que consume mucho tiempo al tener que evaluar todas los casos posibles de valores elegidos para cada hiperparámetro.

3.1.2. Random Search

Al igual que Grid Search definimos un conjunto de valores posibles por cada hiperparámetro. Probamos k combinaciones aleatorias de hiperparámetros y nos quedamos con la mejor. Decidimos cuánto tiempo invertir pero no probamos todas las combinaciones. La desventaja es que quizás nuestro óptimo no sea el más óptimo sino el mejor de los que probamos.

3.2. Random Forest

Es uno de los algoritmos mas populares en clasificación. Tiene buenos resultados para la mayoria de los set de datos pero en general no los mejores. Es un bagging sobre árboles de decision, cada árbol usa un subconjunto de atributos y un bootstrap del set de entrenamiento. Entre los hiperparámetros mas importantes esta la cantidad de árboles y la cantidad de atributos por árbol.

Primero entrenamos el modelo con los hiperparámetros por default. Observamos la importancia que el modelo le daba a los features (Figura 1) y a partir del resultado mejoramos el proceso de Feature Engineering para este modelo.

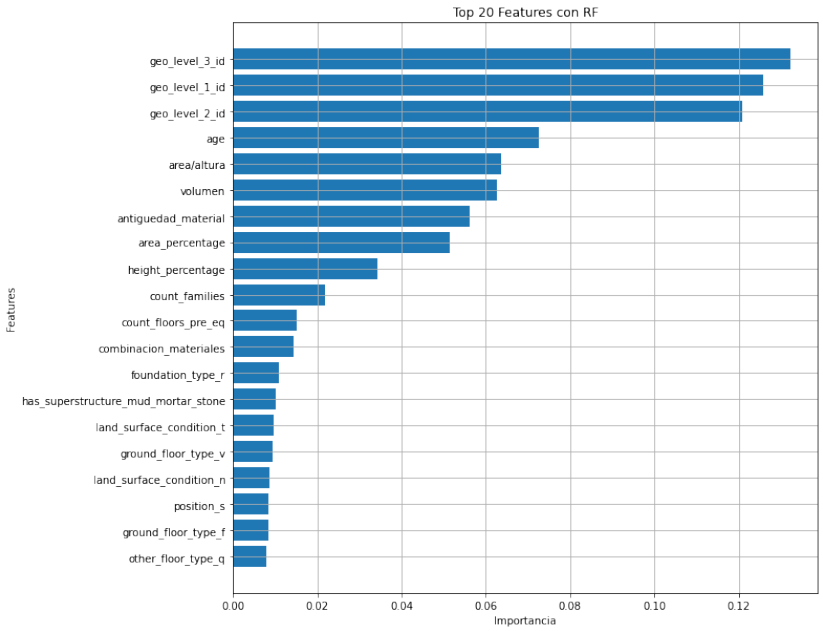

Figura 1: Top 20 Features con Random Forest con hiperparámetros default

Refinamos el modelo a partir de la busqueda de hiperparámetros óptimos. En una primera instancia elegimos Grid Search, fuimos variando los intervalos de los hiperparámetros. Para ello, obtuvimos el grado de importancia que el modelo le estaba dando a los features (Figura 2).

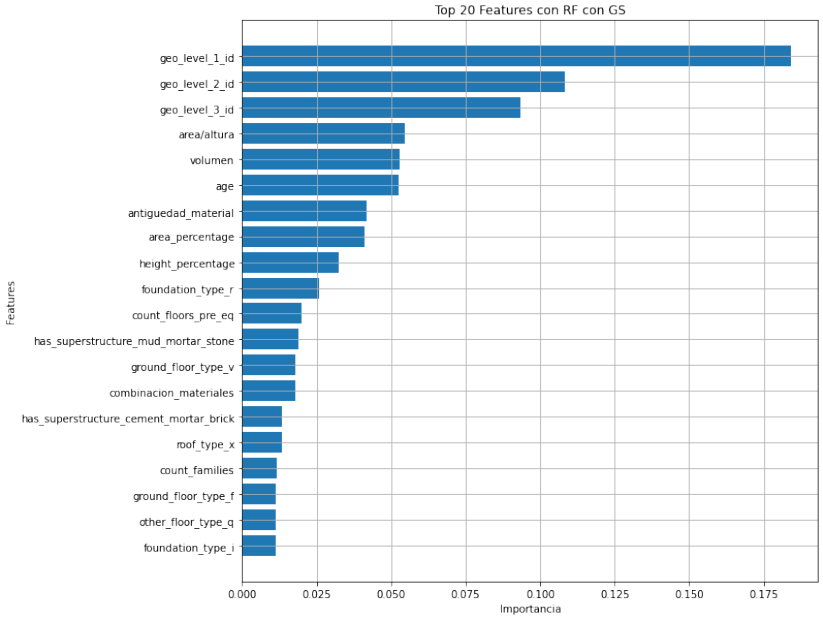

Figura 2: Top 20 Features con Random Forest con Grid Search

Notamos que los features son los mismos, pero hay una diferencia en cuanto a la distribución y magnitudes de la importancia.

Observamos que las variables más relevantes eran las correspondientes a los geo_level_id, la antigüedad y a las características físicas de las edificaciones. Por lo tanto, procedimos incorporando más columnas que relacionan a estos features y obtuvimos su importancia (Figura 3).

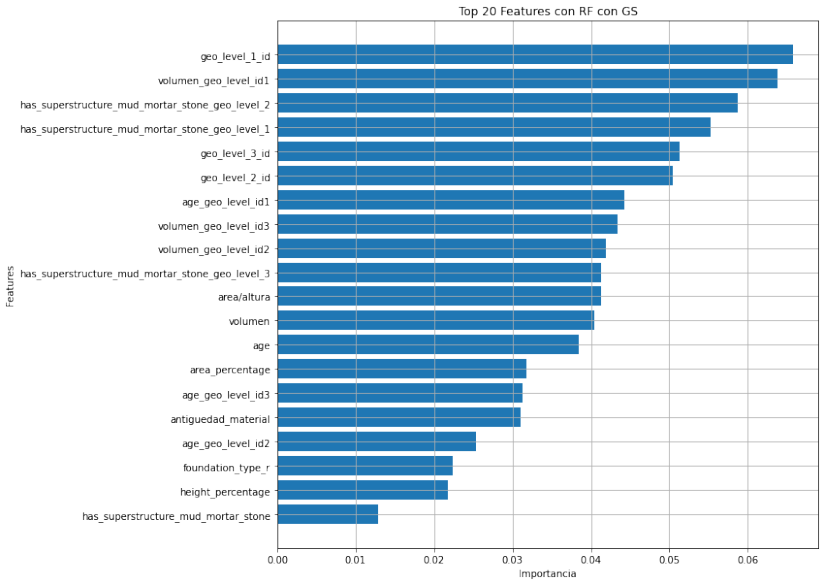

Figura 3: Top 20 Features con Random Forest con Grid Search

Notamos que los features incoporados fueron importantes y el peso de los mismos se redistribuyó generando una mayor equidad entre la relevancia que se les otorga.

3.3. XGBoost

Este algoritmo es un boosting de árboles de decisión. No soporta atributos categóricos. La predicción sobre un feature es la sumatoria de la predicción de varios árboles. Cada árbol nuevo intenta corregir los errores del anterior y son menos profundos que los obtenidos en Random Forest ya que no se enfocan en clasificar bien los datos sino en contribuir para la prediccion final.

Primero entrenamos el modelo con los hiper parámetros por default. Obtuvimos el grado de importancia que el modelo le asignó a los features y nos quedamos con los 20 más relevantes (Figura 4).

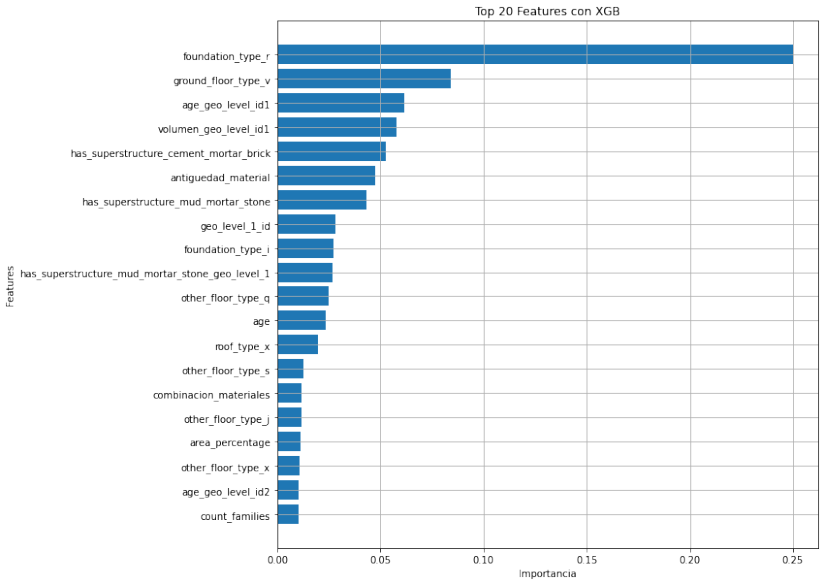

Figura 4: Top 20 Features con XGBoost con hiperparámetros default

Refinamos el modelo a partir de la búsqueda de hiper parámetros óptimos. Para esto utilizamos random search.

Fuimos variando los intervalos de los hiper parámetros hasta quedarnos con los mejores. Finalmente, entrenamos el modelo con estos. Obtuvimos el grado de importancia que el modelo le estaba dando a los features (Figura 5).

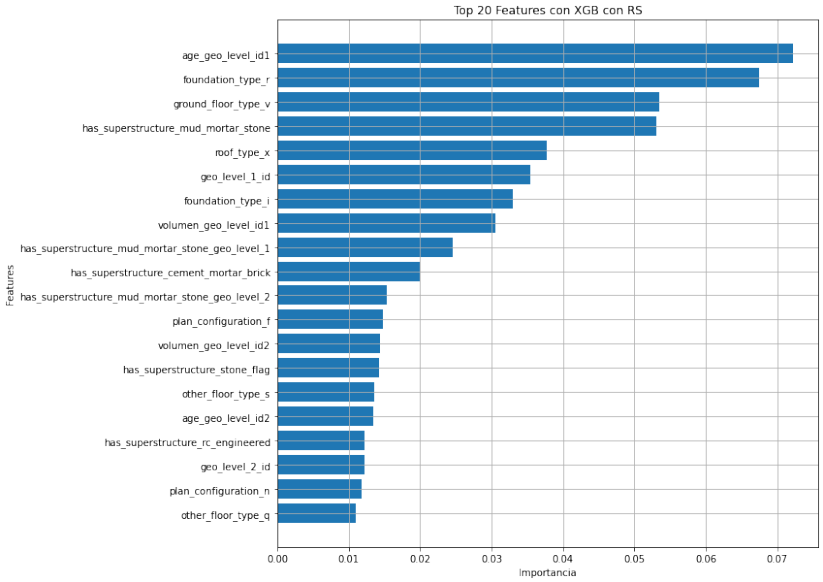

Figura 5: Top 20 Features con XGBoost con Random Search

Observamos que los features más relevantes son distintos a los obtenidos

mediante a Random Forests.

3.4. CatBoost

Este es un algoritmo de gradient boosting que, a diferencia de XGBoost acepta variables categoricas. De ahi su nombre, es una combinación de las palabras 'Category' y 'Boosting'. Con este algoritmo obtuvimos el mejor score.

3.4.1. Default

Inicialmente lo corrimos con los parámetros por default y observamos que resultado nos daba.

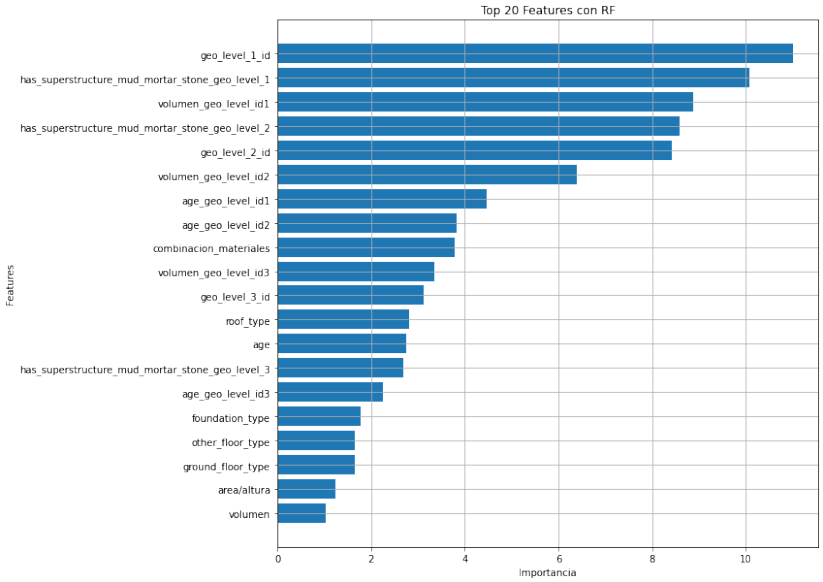

Figura 6: Top 20 Features con CatBoost con default

3.4.2. Grid Search

Luego realizamos la búsqueda de hiperparámetros apartir de Grid Search.

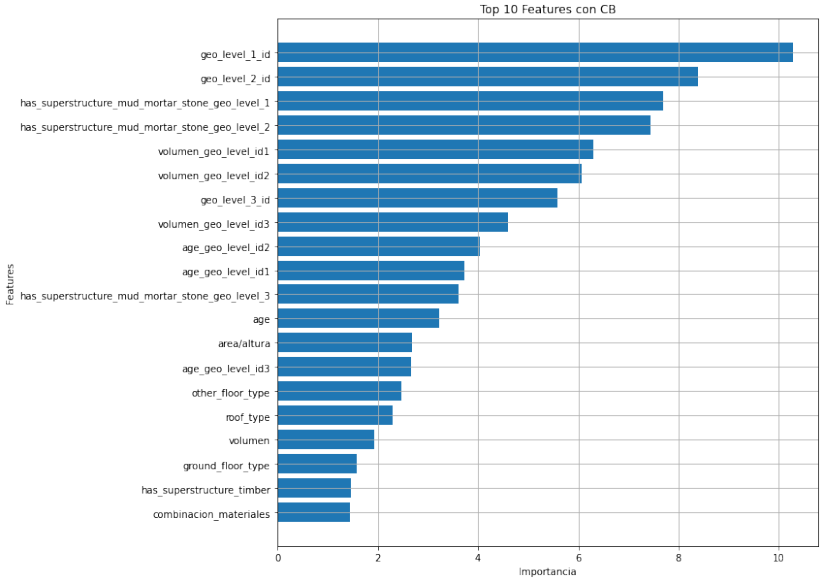

Figura 7: Top 20 Features con CatBoost con Grid Search sin Geo Level Categorical

Después de haber realizado el intento anterior, y viendo que los resultados habían mejorado, transformamos los geo level id a categorical. Esto estaba justificado por la relevalancia que le otorgaba el modelo a los features relacionados con el geo level, y a su vez como el mismo no tenia una relación de orden. Al pasarlos a categorical el modelo podría decidir de qué manera clasificarlos.

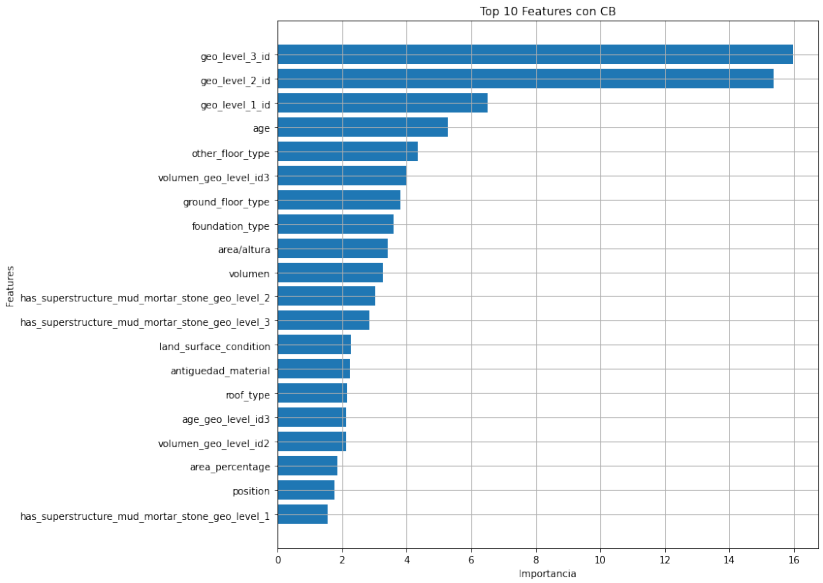

Figura 8: Top 20 Features con CatBoost con Grid Search con Geo Level Categorical

Con este algoritmo obtuvimos nuestro mejor score en Driven Data que fue de 0,7475.

3.5. LightGBM

Es un algoritmo de gradient boosting que usa algoritmos de aprendizaje basados en árboles. Este algoritmo de gradien boosting sobre árboles se diferencia de XGBoost en que construye los árboles segundo las hojas y no segun los niveles.

3.5.1. Default

Inicialmente entrenamos el modelo con los hiperparámetros por defecto para observar que resultados obteniamos. La importancia que le dio este modelo a los features fue la que se observa en (Figura 9).

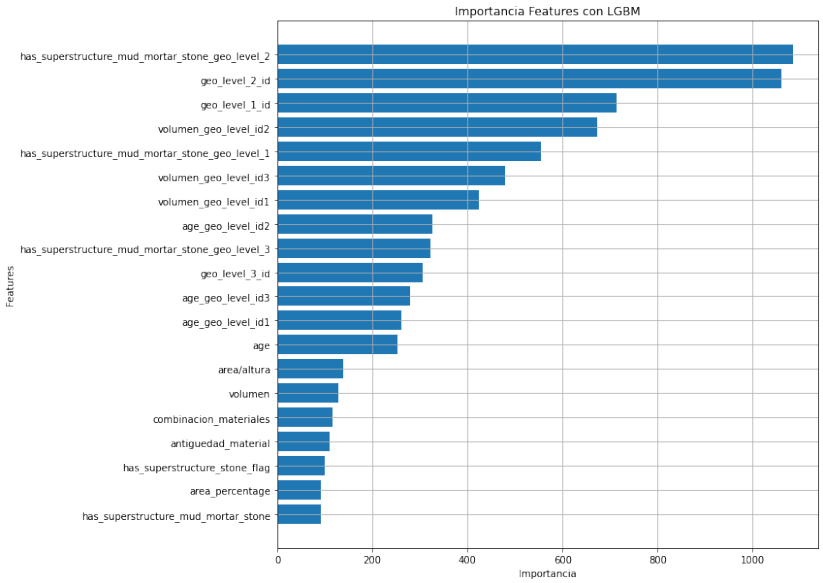

Figura 9: Top 20 Features con LGBM con default

Luego refinamos el modelo a partir de la búsqueda de hiperparámetros óptimos donde utilizamos random search.

3.5.2. Random Search

A pesar de que los resultados con el random search mejoraron, no fueron suficientes para mejorar el comportamiento obtenido con el CatBoost.

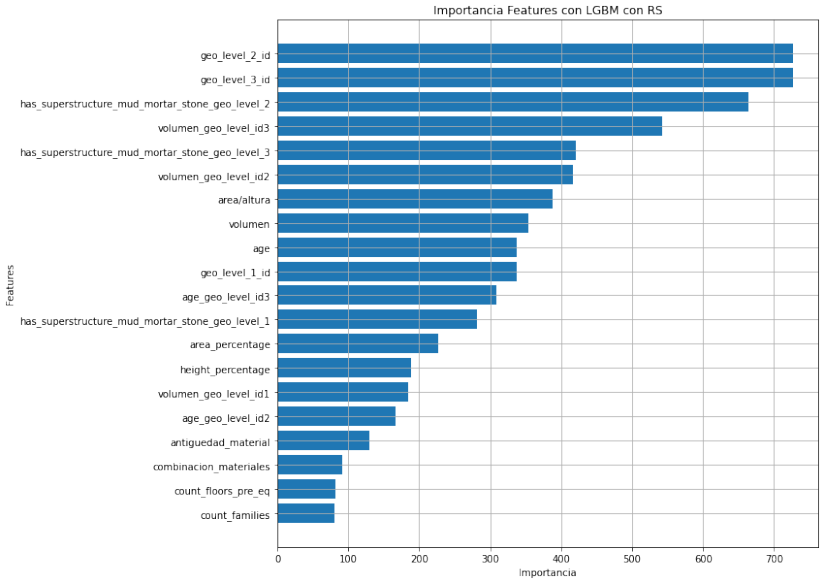

Figura 10: Top 20 Features con LGBM con Random Search

4. Conclusión

En resumen, los modelos de boosting fueron los que presentaron mejores resultados, teniendo en cuenta que el random forest requirió mucho trabajo para conseguir un resultado consireablemente bueno, mientras que los otros en pasadas default ya alcanzaban un buen rendimiento.

Cada modelo le dio diferente importancia a los features, y creemos nosotros que el mejor resultado fue por el comportamiento que le dio catboost a los features categoricos, a diferencia de los otros dos algoritmos de boosting que fueron codificados con one hot encoding para trabajar con dichos features.

Por otro lado cometimos el error de concentrarnos demasiado en el catboost (probando muchos hiperparametros e intentando agregar algún feature que mejore la importancia). Deberiamos haber probado de manera más equitativa los distintos modelos, y no haber estado tanto tiempo intentando mejorar solo uno.