FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Richter's Predictor

[7506] Organización de Datos
Primer cuatrimestre de 2021

Alumno:	LUZZI, Facundo
Número de padrón:	105229
Email:	fluzzi@fi.uba.ar
Alumno:	CASERIO, Juan Cruz
Número de padrón:	104927
Email:	jcaserio@fi.uba.ar

Índice

1. Links

Link de Repositorio de GitHub: https://github.com/juanchycc/Tp2-Datos
Link de Presentación (YouTube):https://youtu.be/vh3s7v1ocUI
Link de Presentación (Drive):https://drive.google.com/file/d/1MM4oTDSm3iZsXLxT-9mFDw9uxbpS0f1R/view

2. Introducción

En este informe, vamos a explicar nuestro enfoque para predecir el daño a la construcción ocurrido durante el terremoto de 2015 en Nepal, India.

3. Análisis Explotatorio de Datos

Para comenzar, vamos a realizar un análisis completo del dataset. Observando cómo se relacionan entre si los distintos features, viendo la correlación entre los mismos. Realizando un pre-procesamiento de los datos.

```
Data columns (total 39 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   geo_level_1_id                          260601 non-null  int64
 1   geo_level_2_id                          260601 non-null  int64
 2   geo_level_3_id                          260601 non-null  int64
 3   count_floors_pre_eq                     260601 non-null  int64
 4   age                                     260601 non-null  int64
 5   area_percentage                         260601 non-null  int64
 6   height_percentage                       260601 non-null  int64
 7   land_surface_condition                  260601 non-null  object
 8   foundation_type                         260601 non-null  object
 9   roof_type                               260601 non-null  object
 10  ground_floor_type                       260601 non-null  object
 11  other_floor_type                        260601 non-null  object
 12  position                                260601 non-null  object
 13  plan_configuration                      260601 non-null  object
 14  has_superstructure_adobe_mud            260601 non-null  int64
 15  has_superstructure_mud_mortar_stone     260601 non-null  int64
 16  has_superstructure_stone_flag           260601 non-null  int64
 17  has_superstructure_cement_mortar_stone  260601 non-null  int64
 18  has_superstructure_mud_mortar_brick     260601 non-null  int64
 19  has_superstructure_cement_mortar_brick  260601 non-null  int64
 20  has_superstructure_timber               260601 non-null  int64
 21  has_superstructure_bamboo               260601 non-null  int64
 22  has_superstructure_rc_non_engineered    260601 non-null  int64
 23  has_superstructure_rc_engineered        260601 non-null  int64
 24  has_superstructure_other                260601 non-null  int64
 25  legal_ownership_status                  260601 non-null  object
 26  count_families                          260601 non-null  int64
 27  has_secondary_use                       260601 non-null  int64
 28  has_secondary_use_agriculture           260601 non-null  int64
 29  has_secondary_use_hotel                 260601 non-null  int64
 30  has_secondary_use_rental                260601 non-null  int64
 31  has_secondary_use_institution           260601 non-null  int64
 32  has_secondary_use_school                260601 non-null  int64
 33  has_secondary_use_industry              260601 non-null  int64
 34  has_secondary_use_health_post           260601 non-null  int64
 35  has_secondary_use_gov_office            260601 non-null  int64
 36  has_secondary_use_use_police            260601 non-null  int64
 37  has_secondary_use_other                 260601 non-null  int64
 38  damage_grade                            260601 non-null  int64
dtypes: int64(31), object(8)
```

Como podemos apreciar, no hay pérdida aparente de datos. No hay valores nulos.

3.1. Features Numéricos

Vamos a realizar un análisis de los Features Numéricos para poder observar como tratar a cada una en nuestro modelo.

3.1.1. Distribución del grado de daño

Para el grado de daño podemos observar un gran desequilibrio entre los datos hacia el grado de nivel 2. Esto puede generar un conflicto a la hora de predecir en el data test, debido a que no tenemos suficiente información para los valores del grado de daño 1 y 3.

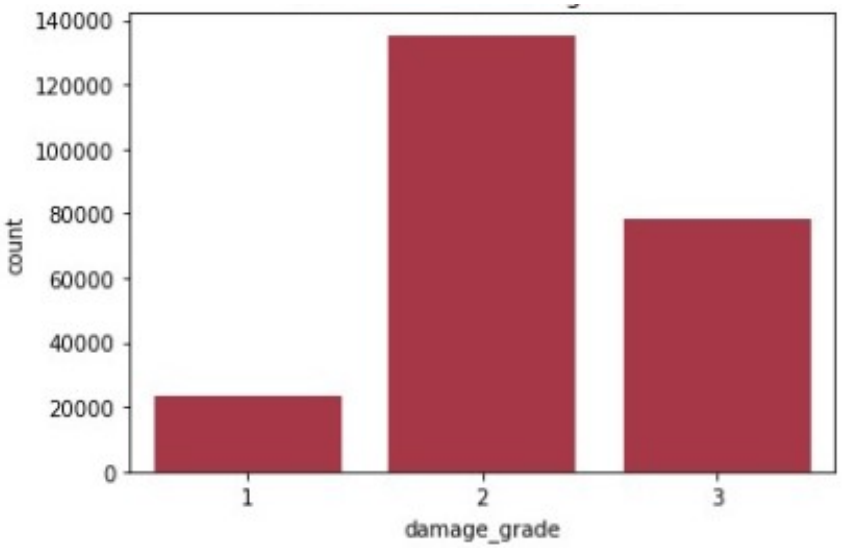


Figura 1: distribución de las variables del damage grade.

3.1.2. Análisis de la Edad

En el caso de la edad de las construcciones, las edades mayores a 100 representan valores atípicos, por lo que vamos a reemplazarlas por el valor promedio.

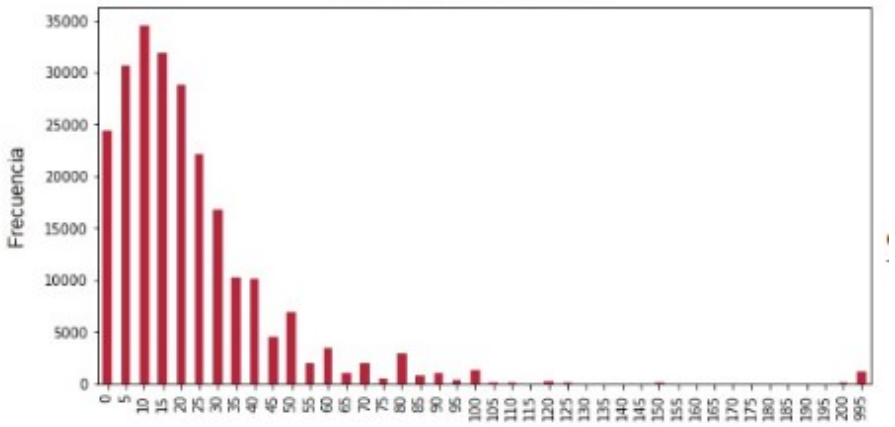


Figura 2: distribución de las variables de la edad.

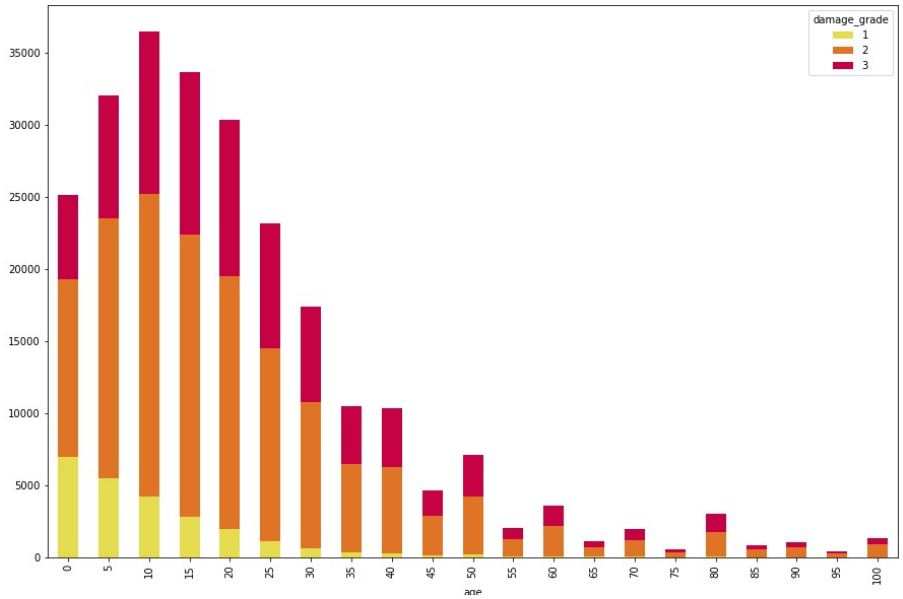


Figura 3: distribución de las variables de la edad.

3.1.3. Distribución según la cantidad de pisos

Podemos observar que la gran mayoría de los afectados fueron los edificios de 2 pisos.

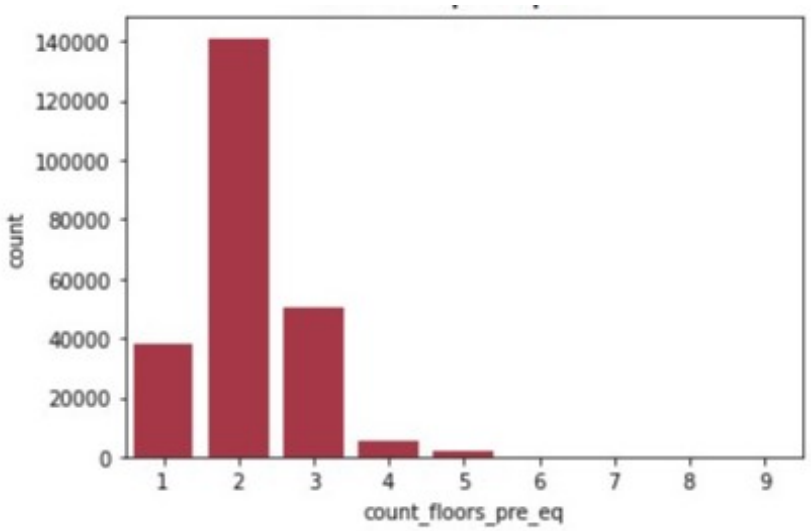


Figura 4: Frecuencia por Cantidad de Pisos.

3.1.4. Tipo de daño según altura de la construcción

En el siguiente gráfico podemos observar como la altura no tiene una correlación directa con el grado de daño. A los valores atípicos (promedio de altura>10) vamos a modificarlos por el promedio.

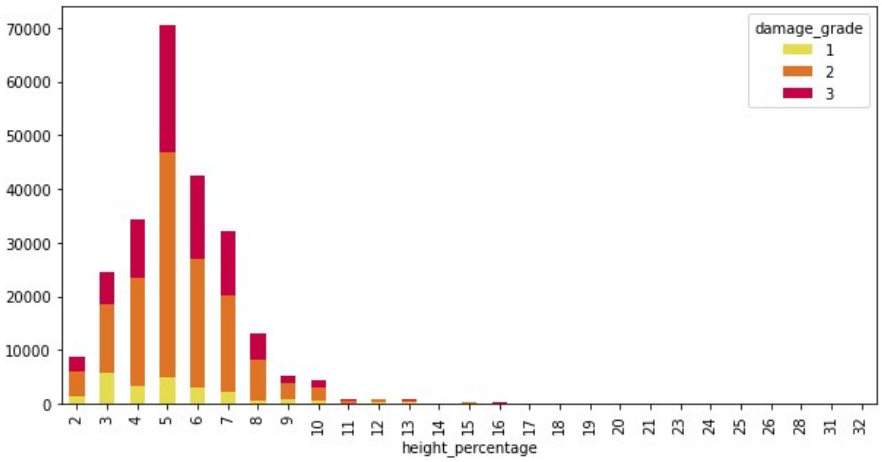


Figura 5: Grado de daño según la altura de la construcción.

3.1.5. Distribución de los valores posibles del área

Notamos que la distribución es muy despareja, por lo que los valores fuera de los más repetidos los reemplazaremos por el valor promedio. La escala fue recortada para el gráfico.

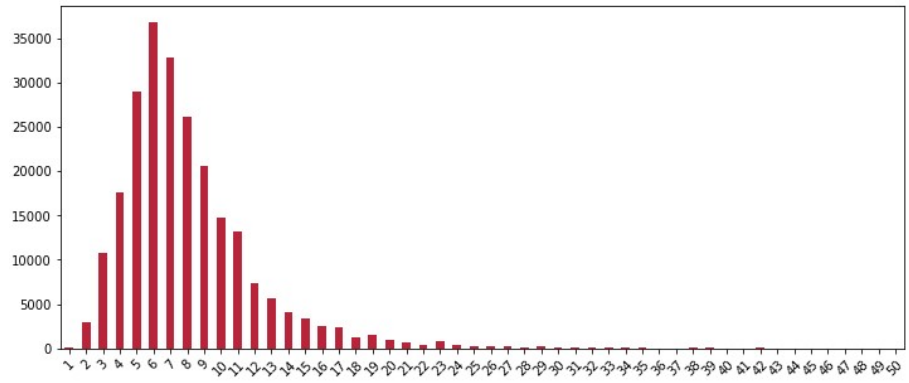


Figura 6: Grado de daño según la altura de la construcción.

3.1.6. Frecuencia por cantidad de familias

En el siguiente gráfico podemos percibir que en la mayor parte de las construcciones solo había una familia. Además, consideramos que el daño no varía en relación a la cantidad de familias. Esto podría indicar que esta columna no tiene nada para aportar a nuestro modelo, por lo que podriamos eliminarla.

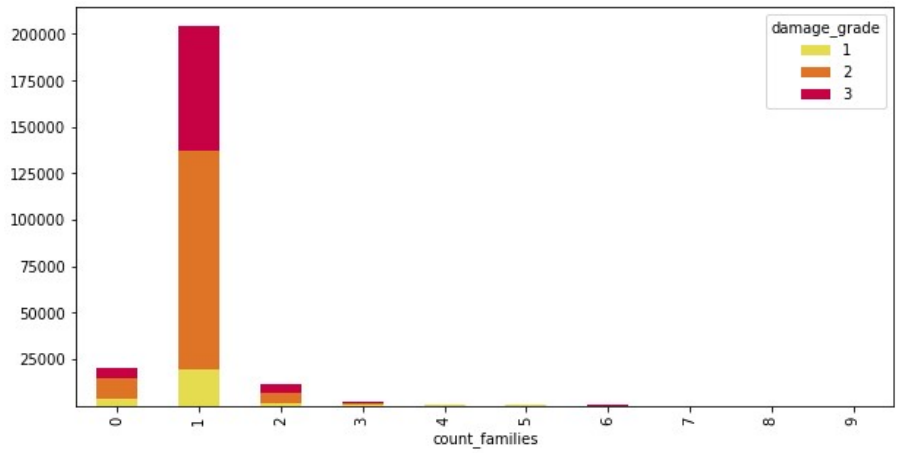


Figura 7: Frecuencia de las variables de la cantidad de familias.

3.1.7. Frecuencia por tipo de uso secundario

Aproximadamente 30.000 construcciones de nuestro set de datos tenían algún uso secundario. Teniendo esto en cuenta, y que los features son columnas binarias compuestas en su mayoria por 0's, consideramos que no aportarán nada a nuestro modelo.

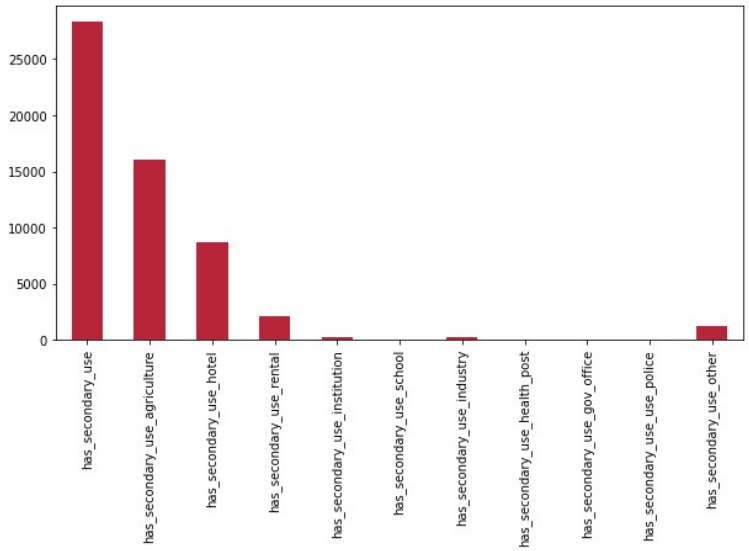


Figura 8: Grado de daño según el tipo de uso secundario de la construcción.

3.1.8. Distribución del daño según el tipo de material

A partir de los features del sig. gráfico, implementaremos un nuevo feature llamado 'resistencia', el cual representara la resistencia de cada construcción, dependiendo el tipo de materiales con los que fue construida. Creemos que la resistencia de un objeto (en este caso una construcción) es un valor que aportará

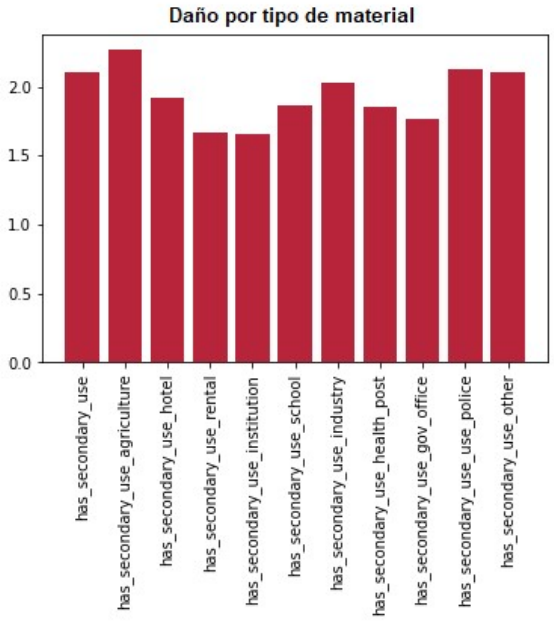
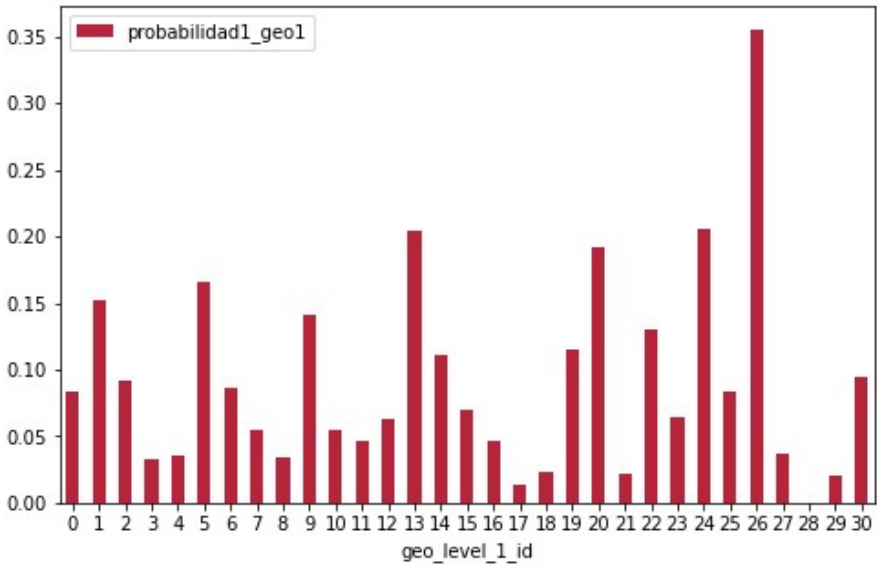
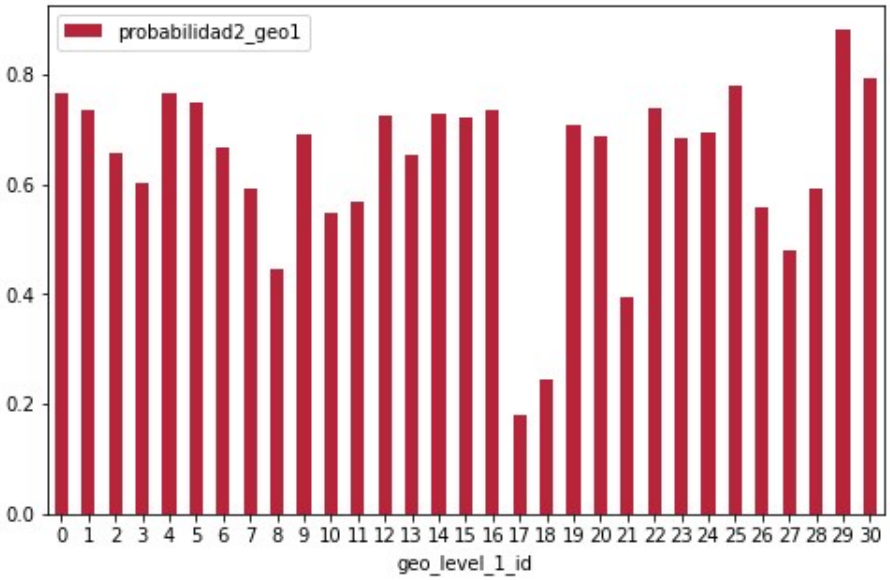


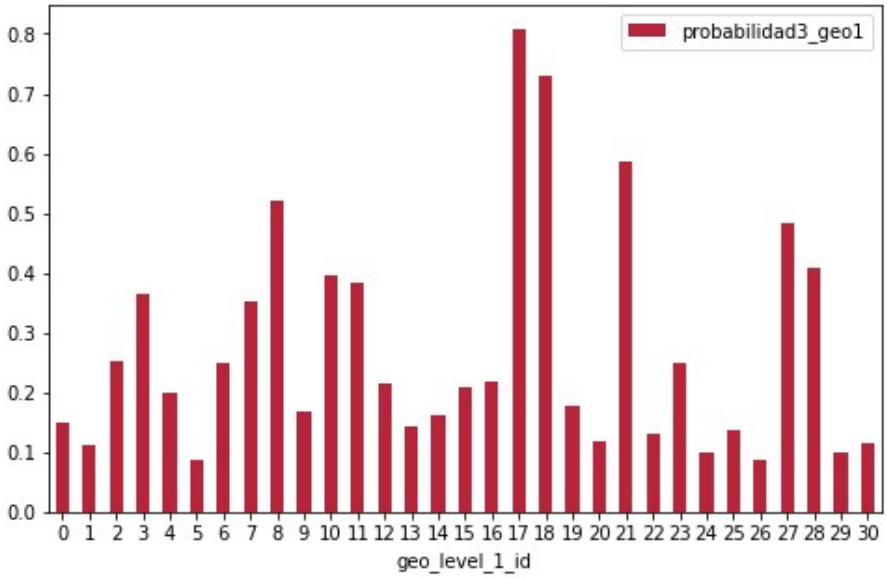
Figura 9: Grado de daño según el tipo de material con el que fue creada la construcción.

3.1.9. Análisis de los distintos Geo Levels

En los siguientes gráficos analizamos la probabilidad de damage grade para cada valor del geo level 1. A simple vista, observando los gráficos podemos ver cuál es el epicentro del terremoto. Por ejemplo, en el Geo Level 17, tenemos valores muy bajos para el grado de daño 1 y 2, y muy altos para el grado de daño 3.







3.2. Features Categóricos

Continuaremos con un análisis en las columnas categóricas.

- **Condición de la superficie de la tierra** valores posibles: n, o, t
- **Tipo de Fundación** valores posibles: h, i, r, u, w
- **Tipo de Techo** valores posibles: n, q, x
- **Tipo de Planta Baja** valores posibles: f, m, v, x, z
- **Otro Tipo de Piso** valores posibles: j, q, s, x
- **Posición** valores posibles: j, o, s, t

- **Configuración del plan** valores posibles: a, c, d, f, m, n, o, p, q, s, u
- **Estado de propiedad legal** valores posibles: a, r, v, w

Para todos estos features, realizamos un proceso de One Hot Encoding, creando tantas columnas binarias como valores posibles tenga el feature.

4. Modelo Inicial - Random Forest

4.0.1. Primeros Pasos

Nuestra primera predicción comenzo con un modelo muy simple de Random Forest, utilizando las columnas que en principio estimamos que son de gran importancia: edad, geo level 1 y 2, porcentaje de area, cantidad de pisos antes del terremoto y porcentaje de altura. En este caso los hiperparametros utilizados fueron los usados por defecto. El submit obtenido fue de 0.6430. Luego se probó eliminando aquellos datos atípicos en las columnas de cantidad de pisos y edad, en el caso del primero hubo una pequeña mejora pero con el segundo no.

4.0.2. Primera búsqueda de hiper parámetros

Lo siguiente que probamos fueron diferentes valores en los hiperparametros, observando un poco el comportamiento y luego utilizando grid search, los hiperparametros buscados fueron: n estimators, max depth y max features. Los resultados no fueron tan buenos como los esperados pero se logro mejorar en un pequeño porcentaje el submit.

4.0.3. Sumando Feature Engineering

Las últimas pruebas realizadas con este modelo fueron las de aplicar diferentes estrategias de Feature Engineering, creamos la columna **Base Condition** la cual se armo utilizando one hot encoding en land surface condition y foundation type luego se multiplico por su promedio a cada columna binaria, y finalmente se sumaron todas.
Por otro lado, para el caso de los featuring de materiales de construcción se observó en un gráfico (figura 9) la cantidad de daño recibido por cada tipo, y de esa forma se determinó una constante de resistencia con la cual se multiplico a cada columna binaria de material, y finalmente se sumaron para conformar la columna **Resistance**.

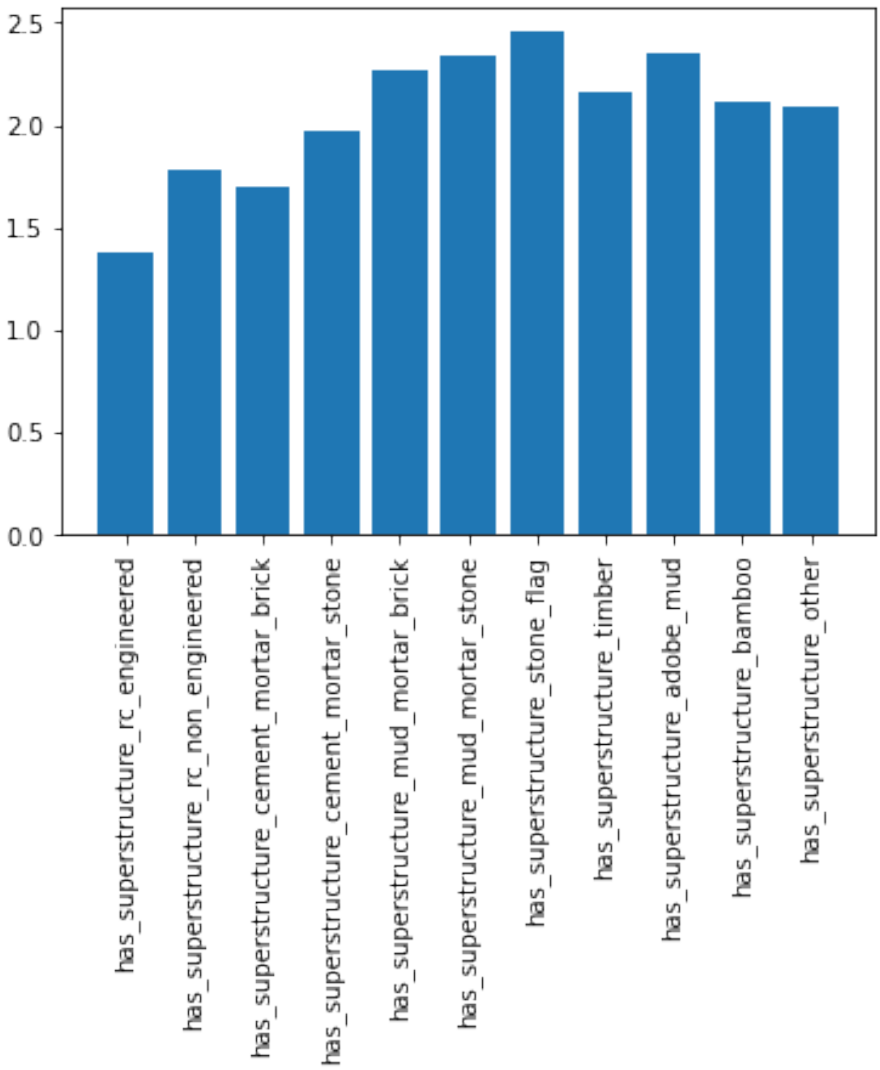


Figura 10: Grado de daño según el tipo de material con el que fue creada la construcción.

Realizando este pequeño featuring se llegó a la mejor predicción obtenida con Random Forest, con una métrica de 0.6921. Cabe aclarar que hasta aca no habiamos realizado un split para test y train por lo que al correr el F1 score no reflejaba un valor semejante al del submit, este paso permitió observar mejor como variaba la predicción de acuerdo a los cambios realizados, el split fue de 0.8 para train y 0.2 para test. En este punto nos pareció que ya era hora de pasar a un modelo que posiblemente funcione mejor en este tipo de pruebas.

5. Modelo Principal - XGBoost

5.0.1. Idea Base

Se partió con la misma base final del modelo de Random Forest, sumándole además las columnas roof type, ground floor type y other floor type, dado que son features categóricos se recurrió al one hot encoding. Además, se múltiplico a cada columna por su promedio, y se sumo con el resto (para cada una de las 3 columndas mencionadas) de esta manera el modelo alcanzo un resultado 0.7233 en Driven Data.

5.0.2. Feature Engineering

A partir de los resultados obtenidos con la idea que ya teniamos, fuimos probando nuevas combinaciones de Features, además de intentar sumar Features creados por nosotros.
A continuación, estaremos explicando con detalle cada uno de ellos:

- **base condition**: la explicación de este feature se puede encontrar en el proceso de Feature Engineering realizado en el modelo de Random Forest.
- **resistance**: la explicación de este feature se puede encontrar en el proceso de Feature Engineering realizado en el modelo de Random Forest.
- **height per floors**: creamos un nuevo feature que consiste en dividir la altura de cada construcción por su cantidad de pisos totales.
- **construction type**: lo creamos teniendo en cuenta los features categoricos que definen el tipo de construcción, y cimientos utilizados cuando se construyó la edificación. El proceso que realizamos fue aplicar one hot encoding a cada uno de los features, y luego sumar el resultado.
- **volume percentage**: armamos este feature multiplicando el valor del área de cada construcción por la altura de la misma.
- **probability of damage grade per geolevel**: calculamos la probabilidad de damage grade para cada nivel de geo level 1, geo level 2 y geo level 3. A partir de eso, creamos 9 features nuevos: probabilidad de damage grade 1 para geo level 1, probabilidad de damage grade 2 para geo level 1, probabilidad de damage grade 3 para geo level 1, ... , probabilidad de damage grade 3 para geo level 3.
- **age mean, height mean, area mean**: para construir estos nuevos features, utilizamos dos combinaciones de features existentes: por un lado, agrupamos por geo level 1, geo level 2 y geo level 3, y para cada variable (age, height, area) calculamos su promedio. De esta misma forma agrupamos foundation type, roof type, ground floor type y other floor type, y calculamos el promedio para cada variable.

Por otra parte, probamos realizar un proceso de eliminación y filtrado de los valores atípicos de los siguientes features: age, count floors pre eq, area percentage, height percentage.

- **Age**: En el caso de la edad, probamos distintas estrategias. En primer lugar, si la edad de un edificio era mayor a 200, la reemplazabamos por el valor promedio. Para el mismo caso, también intentamos eliminar los datos de las construcciones con una edad mayor a 200. Por último, observando que la edad de las construcciones aumentaba de 5 años en 5 años, aplicamos el logaritmo para intentar acotarla.
- **Count Floors Pre Eq**: Para las construcciones con una cantidad de pisos superior a 5, por un lado, tomamos un limite igual a 5. Es decir, los valores mayores a 5 los reemplazamos justamente por 5.
  Por otro lado, nos quedamos con los datos que poseen una cantidad de pisos menor o igual a 5.
- **Area Percentage**: Excluimos del análisis los valores del area percentage mayores a 20.
- **Height Percentage**: Excluimos del análisis los valores del height percentage mayores a 10.

5.0.3. Búsqueda de hiperparámetros

Nuestra estrategia se baso en establecer rangos mínimos y máximos para cada hiperparametro.
n estimators: [100: 1000]
max depth: [6: 12]
learning rate: [0.01, 0.1, 0.3]
colsample bytree: [0.7, 0.8, 0.9, 1.0]
Una vez definidos los intervalos, fuimos probando con Random Search, y acotando cada vez mas los extremos. Nuestro resultado final fue el siguiente:

Buscamos hiperparametros ideales con Random Search

```
n_jobs=[-1]
n_estimators=np.arange(300, 500,100)
colsample_bytree=[0.8, 1.0]
max_depth=np.arange(10, 12, 1)
learning_rate = [0.1]
param_grid={'n_jobs':n_jobs,
            'n_estimators':n_estimators,
            'max_depth':max_depth,
             'learning_rate': learning_rate
            }
```

```
rf=XGBClassifier(eval_metric='mlogloss')
kf=KFold(n_splits=2,shuffle=True)
```

```
rs=RandomizedSearchCV(rf,param_distributions=param_grid,cv=kf,scoring='f1_micro')
```

```
start_time = timer(None)
rs.fit(train_values_subset, train_labels.values.ravel())
timer(start_time)
```

```
 Time taken: 0 hours 36 minutes and 17.93 seconds.
```

```
rs.best_params_
```

```
{'n_jobs': -1, 'n_estimators': 300, 'max_depth': 10, 'learning_rate': 0.1}
```

Figura 11: Valor de Hiperparametros finales, hallados con Random Search.

5.0.4. Resultados en Driven Data

Idea Base
Nuestra primera aproximación con el modelo XGBoost, probando con los mismos Features de que utilizamos en el Random Forest, y sumándole como ya comentamos anteriormente los Features Categoricos transformados previamente, logramos mejorar nuestra predicción llegando a un submit de **0.7233.**

0.5919 Decidimos probar excluir del análisis algunos valores de el area y height percentage. Esto nos dio un pésimo resultado, por lo que decidimos ir para atras con esta decisión, y mantener los valores originales.

0.5835 Reemplazamos los valores de la edad mayores a 200, por el valor promedio, y para la cantidad de pisos, establecimos un limite igual a 5. Nuevamente, nos dio un pésimo resultado, por lo que por el momento decidimos dejar de innovar con esas variables.

0.7257 Agregamos el feature Height per Floor, dividiendo la altura por la cantidad de pisos. También, teniendo en cuenta que la correlación entre las variables height percentage y count floors pre eq, decidimos eliminar una de ellas, height percentage.

0.7266 Incorporamos el feature Volume Percentage.

0.7271 Como comentamos previamente, eliminamos height percentage debido a la alta correlación que tenia con otro feature. Probamos agregarlo, y ver cual era el resultado, y nos sorprendió ver que mejoró.

0.7434 Nos dimos cuenta que estabamos buscando incorporar nuevos features, pero en ningún momento probamos con todos los ya existentes. Error de nuestra parte.
Para este resultado, utilizamos absolutamente todos los features, codificando aquellos categoricos con One Hot Encoding, y eliminando count families, y legal ownership status, por que consideramos que no aportaría mucho al modelo conocer la cantidad de familias que vivían en una construcción, y por otro lado, tampoco aportaría saber el estado legal de la tierra en la que se encontraba la edificación.

0.7349 A partir del resultado obtenido en el paso anterior, fuimos probando incorporar los nuevos features que creamos nosotros. Nos dimos cuenta que la resistencia de la construcción, que anteriormente nos había dado buenos resultados, ahora nos tira para abajo.

0.7436 Agregamos count families.

0.7448 Agregamos Base Condition y Volume Percentage

0.7433 Agregamos Height Per Floor, y legal ownership status. Como esperabamos por parte de legal ownership status, esto no dio un buen resultado. Por otra parte, Height Per Floor que antes mencionado nos había dado un buen resultado, ahora no aporta a la predicción.

0.7459 Agregamos la probabilidad de damage grade para cada nivel de geo level 1, geo level 2, y geo level 3.

0.7446 Incorporamos construction type, age mean, area mean, height mean.

0.7467 Eliminamos construction type, y probabilidad de damage grade por geo level.

0.7479 máx. Decidimos buscar los hiperparametros ideales utilizando Random Search. Obteniendo nuestro mejor score hasta el momento.

5.1. Conclusiones

5.1.1. Random Forest

Al comenzar el trabajo, no pensamos que con un modelo básico, y poco trabajo de feature engineering se pueda obtener un resultado bastante decente como fue en este caso. Luego, como fue de esperar en este modelo al agregar columnas nuevas se obtuvieron mejores resultados. La búsqueda de hiper parámetros no fue de lo más amigable ya que muchas veces se alejaba bastante de resultados obtenidos con hiperparametros buscados a mano, hasta utilizando Grid Search.
Random Forest cumplió e incluso superó las expectativas, y nos encamino para el siguiente modelo. Como ya teniamos esta base fue más fácil la creación de nuevos features teniendo una noción de que podía funcionar, y lo que no.

5.1.2. XGBoost

Como era de esperar, solo realizando el cambio de modelo la predicción sufrió notables mejoras (aunque también mayor esfuerzo de computo). En este punto solo restaba pensar que features iban a ser de utilidad y probar cuales arrojaban los mejores resultados. Sin embargo, esta fué la parte más dificil del trabajo, ya que, no era posible predecir que iba a pasar y todo había que probarlo. La gran sorpresa se nos dio al utilizar mucha mayor cantidad de columnas, si bien el miedo a overfittear siempre estaba tratamos de utilizar un número lógico teniendo en cuenta también que el set de datos cuenta con aproximadamente 260.000 filas. En la última etapa las mejoras eran cada vez de menor porcentaje y las combinaciones a probar cada vez mayores. Por esta misma razón el

mayor desafio no fue sólo encontrar los mejores features, sino también la mejor combinación entre estos.

5.1.3. Feature Engineering

Al arrancar con este trabajo, considerabamos que algunos Features no iban a importar el modelo. Nos preguntabamos
¿Qué podría aportar a una predicción del daño la cantidad de familias que vive en una construcción? ¿Y el uso secundario qué se le daba a la construcción? Ni hablar del estado legal de la tierra donde fue construída la edificación.
Al ir probando todos los features, combinaciones y demás, nos dimos cuenta que arrancamos viendolo desde un enfoque incorrecto. Pudimos concluir que los mejores resultados se dieron realizando muchas pruebas con diferentes combinaciones, y viendo entonces cuál resultaba ser la mejor. En nuestro caso, por ejemplo, habiamos eliminado el feature count families. Luego probamos agregandola, y contrario a lo que pensabamos, nuestra predicción mejoró. Lo mismo ocurrió con los features binarios de uso secundario.

5.1.4. Nuestro mejor resultado

Luego de un arduo trabajo de Feature Engineering, probando diferentes modelos de Árboles de decisión, concluímos que el mejor resultado que obtuvimos fue con XGBoost, obteniendo un submit de 0.7479.
A este resultado llegamos combinando todos los features iniciales, aplicando One Hot Encoding a los features categoricos, y sumando los siguientes features creados por nosotros: base condition, volume percentage, promedio de edad, promedio de altura y promedio de area. Utilizamos siguientes hiperparametros, hallados con Random Search: 'n jobs': -1, 'n estimators': 300, 'max depth': 10, 'learning rate': 0.1