75.06 Organización de datos

Trabajo Práctico N°2: Predicción de la variable "damage grade"

Padrón	Alumno	Dirección de correo
94194	Bravo Reyes,Christian	cbravor@fi.uba.ar
102831	De Feo, Laura	ldefeo@fi.uba.ar
102696	Meza Boeykens, Damian	dmeza@fi.uba.ar
102896	Movia, Guido	gmovia@fi.uba.ar

Índice

1. Introducción

En el trabajo práctico anterior, hicimos un análisis exploratorio sobre los datos recopilados de las edificaciones afectadas por el terremoto Gorkha. Utilizamos el dataset provisto por la organización DrivenData **train_values.csv**, **test_values.csv** y pudimos obtener distintas conclusiones.

Figura 1: Daños por el terremoto

En esta ocasión debemos predecir la variable *damage_grade*, como parte de la competencia de la organización Driven Data. Esta predicción la haremos a través de un modelo de Machine Learning, en el presente escrito tendremos como objetivo enumerar y describir los diversos pasos que conforman el mismo. Ahora bien, ¿cuál es el punto de inicio que debemos tomar para saber si vamos en el camino correcto?. Como punto de inicio tomaremos el modelo entrenamiento y predicción asignado en el tutorial benchmark de la competencia el cual nos da una propuesta de selección de caracteristicas, nos propone la codificación One Hot Encoding y la métrica F1_score que usaremos para la predicción. Ejecutando el código propuesto alcanzamos el score de 0.5815. Este score será nuestro punto de inicio (sabemos que nuestras predicciones deben alcanzar un score superior). Sin más preambulos comenzaremos con el desarrollo de nuestro Machine Learning, en el mismo trataremos de repondernos ¿cuál es el mejor modelo posible?, ¿qué codificación es la adecuada?, ¿qué características nuevas ayudarán a mejorar las predicciones?

2. Desarrollo

Como hemos mencionado, para generar una predicción de la variable 'damage grade' codificaremos un modelo de Machine Learning, pero más importante será trabajar y "moldear"los datos para optimizar la etapa de entrenamiento y así mejorar la predicción. En

esta tarea intervendran diversos pasos entre ellos tendremos: la ingenieria de caracteristicas (Feature Engineering), la elección del modelo con el con la cual realizaremos el entrenamiento, los hiperparametros seleccionados y finalmente la elección de la métrica con la cual lograremos obtener el puntaje que ha alcanzado nuestra predicción.

2.1. Selección de caracteristicas

En nuestro análisis realizado en el informe anterior, pudimos concluir que las variables que influyeron directamente en los daños de las edificaciones son las siguientes:

1. **Foundation Type:** el tipo de cimientos utilizados de los cuales el mas dañino fue el de tipo r, y el menos influyente, el de tipo i.
2. **Ground Floor Type:** tipo de construcción usado en la planta baja cuando se construyó la edificación. El que tuvo menor resistencia al terremoto fue el de tipo f, y el mas resistente, el de tipo v.
3. **Materiales:** esta variable fue una de las mas relevantes a la hora de analizar el daño que recibieron las edificaciones.Se pudo demostrar con los datos utilizados que las estructuras de barro y piedra son las mas fragiles, siendo estas tambien las mas economicas; mientras que, los edificios construidos con cemento y ladrillo fueron los que mas resistieron el golpe.
4. **Roof Type:** tipo de techo usado cuando se construyó la edificación.

.

Por otro lado, nombramos aquellas caracteristicas que, en nuestra opinion, no son relevantes acerca del daño que sufrieron los edificios luego del terremoto:

1. Legal Ownership Status: Estado legal de la propiedad donde la edificación fue construida.
2. Count Families: número de familias que viven en la edificación.
3. Has Secondary Use Other: si el edificio era usado con otro uso secundario. Esta tres variables no guardan relación alguna con la construcción de los edificios.

Estas hipotesis serán comprobadas cuando entrenemos nuestros modelos.

2.2. Feature Engineering (Ingeniería de características)

Probablemente este sea el paso más importante para lograr un adecuado entrenamiento de nuestro modelo. Mientras más creatividad apliquemos al procesamiento de nuestros datos, mejores predicciones obtendremos. Si bien las combinaciones son infinitas, usaremos una lógica intuitiva y nos basaremos en el primer informe entregado, para generar nuevas características que mejoren nuestros modelos. Luego comprobaremos si nuestra intuición fue la correcta calculando los distintos valores del F1_score.

2.2.1. Codificación

Para poder entrenar nuestro modelo es necesario codificar las características (columnas) no numéricas, ya que el proceso requiere valores numéricos. Para esto usaremos principalmente One Hot Encoding, el cual genera por cada valor del feature, una columna binaria; obteniedo así vectores canónicos que representan el valor del feature. Esta codificación produce buenos resultados, pero también tiene sus desventajas.

#	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

Figura 2: Fragmento de columnas del dataset antes de aplicar la codificación

#	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

Figura 3: Fragmento de columnas del dataset después de aplicar la codificación

Como podemos observar en las figuras 2 y 3, al aplicar One Hot Encoding se agrega gran cantidad de columnas nuevas y, si el feature no aporta mucho a la predicción, esto generará *ruido* para el proceso de entrenamiento.

Para resolver esto, aplicaremos otra codificación específicamente para el feature *plan configuration*. Utilizando Mean Encoding podemos traducir cada valor del feature en una ponderación numérica respecto del daño recibido, generando así un solo feature que presenta una proporción entre los daños leves y los extremos.

2.2.2. Características generadas en base a los datos de construcción

Como bien mencionamos en la sección anterior las características *Tipo De Cimientos*, *Tipo De Techo*, *Tipos De Piso PB* y todos los *Materiales presentes en la edificación* presentan una "distancia"que podemos aprovechar para mejorar las predicciones de nuestros modelos. Proponemos entonces el concepto de vulnerabilidad en la cual asociamos un valor numérico resultado de una combinación lineal de coeficientes determinados, tomando el cero como referencia de una edificación regular, valores positivos para edificaciones vulnerables y valores negativos para aquellas más resistentes. Podemos expresar la vulnerabilidad de la siguiente forma:

$$Vulnerabilidad = 1 + \sum_{n=1}^{cant} coef_n * tipo_n \tag{1}$$

Donde $coef_n$ es el peso ponderado del tipo, y $tipo_n$ es el estado binario (si la edificación lo tiene o no)

1. **Cálculo de la vulnerabilidad de los cimientos:** como pudimos observar en el análisis previo, el tipo de cimientos tiene una influencia directa en el daño de las edificaciones (Fig. 4). En este caso que el tipo I tiene una gran resistencia y el tipo R muy poca, por lo tanto - considerando la edificación regular (el cero) entre el tipo R y el U - aplicamos los siguientes pesos (Fig. 5):

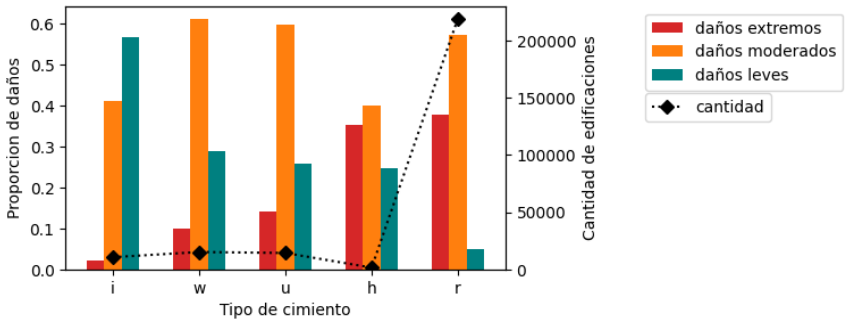

Figura 4: Gráfico de el daño en los edificios según el tipo de cimiento

	peso
foundation_type_h	-3.5
foundation_type_i	-2.5
foundation_type_r	-1.2
foundation_type_u	0.2
foundation_type_w	3.7

Figura 5: Pesos para el cálculo de la vulnerabilidad de los cimientos

2. **Cálculo de la vulnerabilidad de los techos:**

	peso
roof_type_n	3.8
roof_type_q	3.5
roof_type_x	-2.5

Figura 6: Pesos para el cálculo de la vulnerabilidad de los techos

3. **Cálculo de la vulnerabilidad de los distintos tipos de piso en planta baja:**

	peso
ground_floor_type_f	-2.8
ground_floor_type_m	-1.5
ground_floor_type_v	-0.8
ground_floor_type_x	2.5
ground_floor_type_z	3.8

Figura 7: Pesos para el cálculo de la vulnerabilidad de los distintos tipos de piso en planta baja

4. **Cálculo de la vulnerabilidad de los distintos usos de los edificios:**

	peso
has_secondary_use_agriculture	3.3
has_secondary_use_hotel	3.2
has_secondary_use_rental	2.5
has_secondary_use_institution	2.2
has_secondary_use_school	1.0
has_secondary_use_industry	-1.0
has_secondary_use_health_post	-1.8
has_secondary_use_gov_office	-2.5
has_secondary_use_use_police	-3.2
has_secondary_use_other	-3.4

Figura 8: Pesos para el cálculo de la vulnerabilidad de los distintos usos de los edificios

5. **Cálculo de la vulnerabilidad de los materiales:**

	peso
has_superstructure_adobe_mud	4.8
has_superstructure_mud_mortar_stone	3.8
has_superstructure_stone_flag	3.8
has_superstructure_cement_mortar_stone	2.8
has_superstructure_mud_mortar_brick	1.4
has_superstructure_cement_mortar_brick	8.0
has_superstructure_timber	-8.0
has_superstructure_bamboo	-1.6
has_superstructure_rc_non_engineered	-2.4
has_superstructure_rc_engineered	-3.2
has_superstructure_other	-3.6

Figura 9: Pesos para el cálculo de la vulnerabilidad de los materiales

La vulnerabilidad de cimientos, techos, pisos, materiales, son nuevas variables que se generaron a partir del DataSet **train_values.csv** luego de aplicar la codificación, donde contabamos con nuevas caracteristicas (columnas). En cada una de las funciones que generamos con estos nuevos datos, se le otorgo a las variables involucradas un peso el cual no fue al azar, sino que lo aproximamos utilizando los graficos de nuestro primer informe.

2.2.3. Caracteristicas generadas en base a los datos sociales relacionado con la edificación

1. calcularVulTierra

2. calcularVulUso
3. calcularVulEstado
4. calcularVulPlan

2.2.4. Caracteristicas ineficaces al predecir

Al momento de la seleccion de las caracteristicas, notamos que algunas de ellas no fueron muy eficientes a la hora de la prediccion con los distintos modelos utilizados. Podemos ver que seleccionando solo los features ***'land surface condition'***, ***'plan configuration'*** y ***'position'***, independientemente del modelo aplicado, no suman cambios optimos en el proceso, sino que restan significativamente provocando que el error calculado, utilizando la métrica **F1 score micro**, decaiga con brutalidad. Esto lo apreciamos, por ejemplo, utilizando el modelo KNN donde prediciendo con un dataset casi completo obtenemos un error de aproximadamente 0.7250 o mas; en cambio, si aplicamos un feature seleccionando solo las tres mencionadas, obtenemos un valor de aproximadamente 0.4148.

2.3. Modelos predictivos

2.3.1. KNN

El primer modelo predictivo seleccionado es el denominado KNN , el cual permite realizar una predicción para una determinada consulta basandose en los k puntos más cercanos (vecinos más cercanos). El algoritmo de KNN consta de dos hiperparametros que debemos definir , los cuales son la metrica que se utiliza para calcular la distancia entre los puntos y el valor de k que define la cantidad de vecinos que vamos a utilizar. Para poder encontrar los hiperparametros optimos utilizaremos en primera instancia el metodo de Random Search, el cual selecciona de manera aleatoria una cantidad de combinaciones de hiperparametros dentro de un rango establecido y a partir de ello encuentra la combinación más optima. Repetimos este procedimiento una cierta cantidad de veces para poder obtener los candidatos a la combinación más optima, y a partir de ello utilizaremos el metodo de Grid Search el cuál nos permitira encontrar los hiperparametros más optimos de la muestra establecida. Realizando este procedimiento los hiperparametros obtenidos son k = 8 y p = 1 , y si realizamos un grafico para los diferentes valores de k y p en un rango establecido (0 y 50 para ambos casos) , se puede observar que lo obtenido experimentalmente es consistente , ya que en ambos casos el F1 Score adquiere sus valores maximos para los valores seleccionados de k y p.

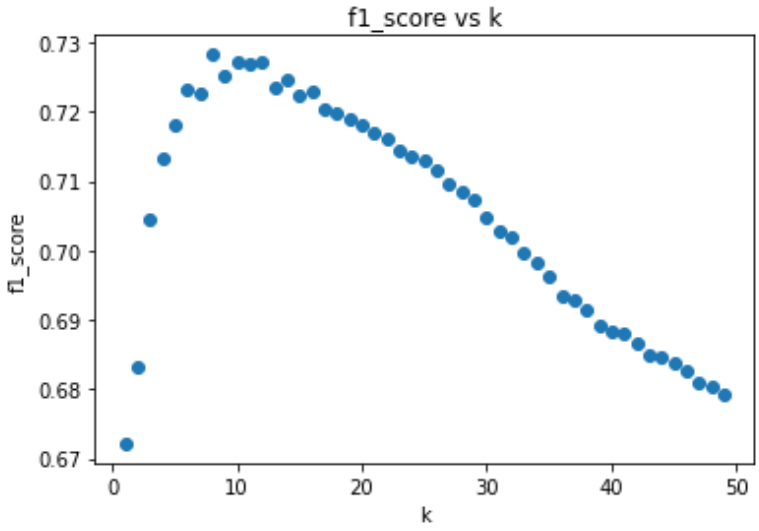

Figura 10: F1 Score vs K

Podemos notar en el grafico de F1 Score vs K que la funcion discreta es creciente hasta el valor k = 8 , adquiriendo su pico maximo , y luego decrece constantemente hasta el ultimo valor posible de k.Podemos observar que los resultados obtenidos experimentalmente coinciden con los argumentos teoricos , ya que si consideramos un numero elevado de k lo que sucedera es que tomaremos una muestra grande de vecinos y por lo tanto se perderan ciertas regiones especificas y se tendera a favorecer a las clasificaciones mas genericas , ya que se dara mayor peso a las clases que tienen mayor cantidad de puntos en el set de entrenamiento.En el caso de tomar valores pequeños de k , como por ejemplo k = 1 lo que sucedera es que el algoritmo tomara el vecino mas cercano y por lo tanto no podra generalizar correctamente , generando problemas de overfitting.

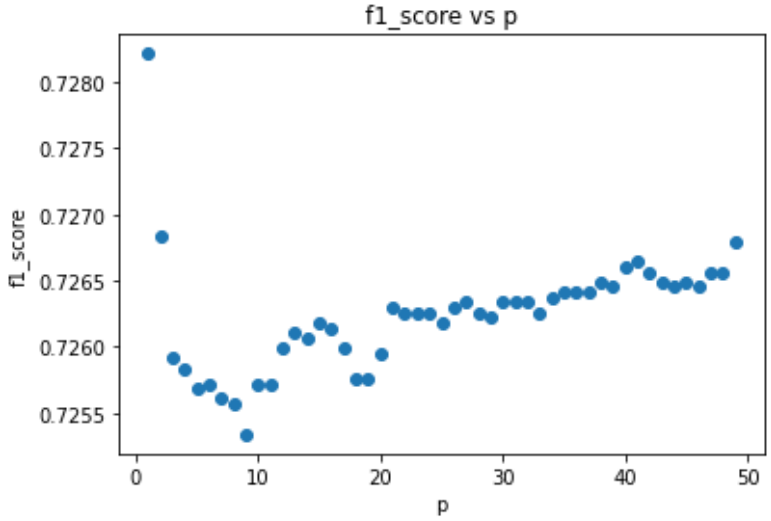

Figura 11: F1 Score vs P

En el caso de la grafica F1 Score vs P se puede observar que tenemos dos puntos que poseen los valores picos , y son los puntos P = 1 y P = 2 , siendo el primero el que define a la distancia Manhattan y el segundo a la distancia Euclideana.Se puede observar que el F1 Score toma su pico maximo si utilizamos la distancia Manhattan , y desarrolla una tendencia ascendente hasta el punto observado en k = 50.

2.3.2. XGBoost

El principio de Boosting nos permite construir un algoritmo preciso a partir de un conjunto de algoritmos que representan modelos simples de prediccion.El modelo XGBoost es un modelo predictivo que utiliza el principio de Boosting y es una implementacion del modelo Gradient Boosting , el cual esta formado por un conjunto de arboles de decision individuales que se entrenan de forma secuencial , de forma tal que cada arbol nuevo trata de mejorar los errores de los arboles anteriores y de esta forma se itera hasta que el error ya no pueda ser corregido.

Si realizamos un esquema del algoritmo podemos concluir que :

1. Se genera un arbol inicial que denominaremos A0 y se busca predecir una variable y , siendo el error de este modelo E0.
2. Se genera un arbol que denominaremos C0 que ajusta el error del arbol A0
3. Se combina el arbol inicial A0 y el arbol corrector C0 y se genera un nuevo arbol A1 , que posee un error cuadratico medio menor al arbol inicial.
4. Se iteran los pasos anteriores hasta que el error es minimo y por ende ya no puede ser optimizado.

Introducido el marco teorico del modelo Xgboost comenzamos con el entrenamiento del mismo para eso vamos a relizar la siguiente selección de características, según las importancias de las mismas A continuación se muestra la tabla de las importancias según el modelo (figura 12)

	Importancia	feature
21	0.000000	has_secondary_use_institution
22	0.000000	has_secondary_use_school
23	0.000000	has_secondary_use_industry
24	0.000000	has_secondary_use_health_post
25	0.000000	has_secondary_use_gov_office
26	0.000000	has_secondary_use_use_police
20	0.006587	has_secondary_use_rental
15	0.008269	has_superstructure_rc_non_engineered
2	0.008723	geo_level_3_id
5	0.008737	height_percentage

Figura 12: Importancia vs Caracteristica

	Sumatoria
has_secondary_use_use_police	23.0
has_secondary_use_gov_office	38.0
has_secondary_use_health_post	49.0
has_secondary_use_school	94.0
has_secondary_use_institution	245.0
has_secondary_use_industry	279.0
has_secondary_use_other	1334.0
has_secondary_use_rental	2111.0
has_superstructure_other	3905.0
has_superstructure_rc_engineered	4133.0
has_superstructure_cement_mortar_stone	4752.0

Figura 13: Features vs Sumatorias

Para buscar los hiperparametros más optimos para maximar el F1_Score graficamos

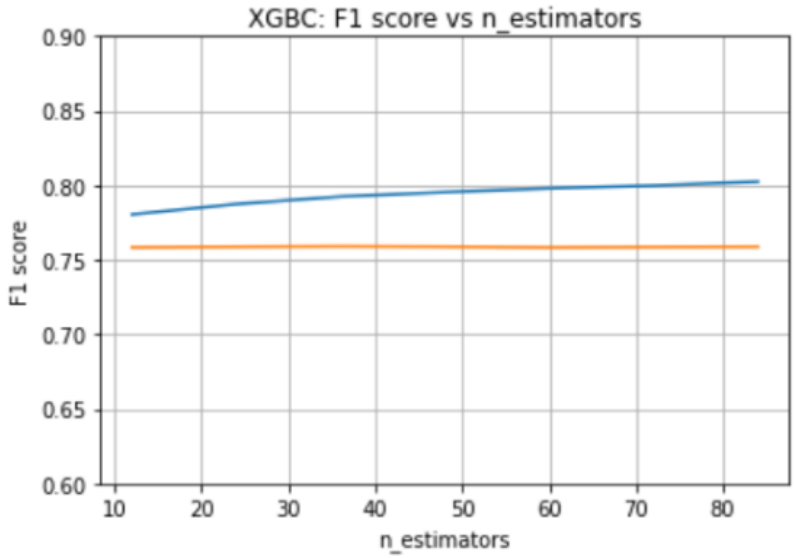

Figura 14: F1_Scores vs n_estimator

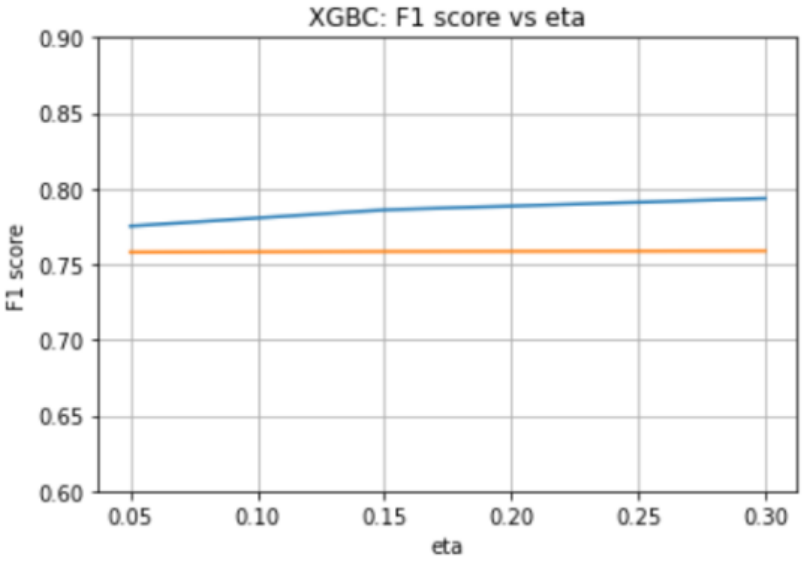

Figura 15: F1_Scores vs Eta

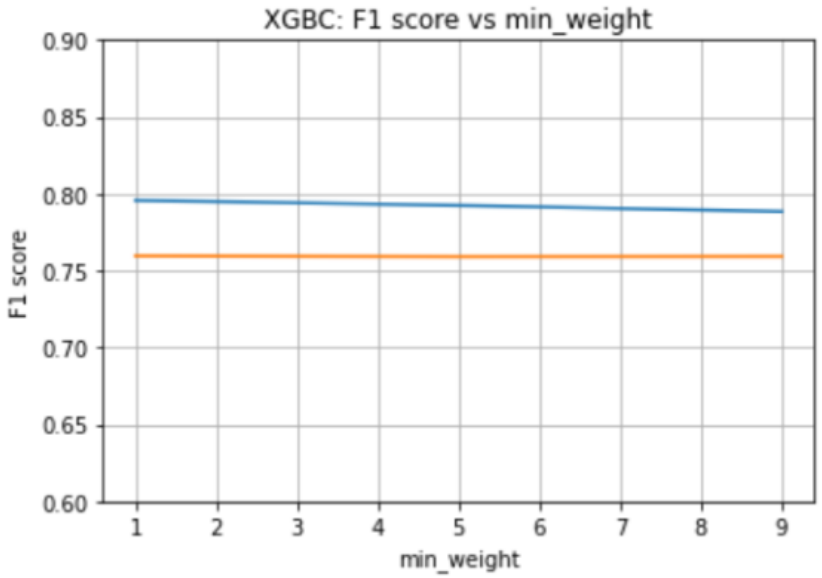

Figura 16: F1_Scores vs Min_weight

2.3.3. Random Forest

El random Forest consiste en un conjunto de arboles de decision combinados utilizando la tecnica de bagging. Esta tecnica consiste en crear diferentes modelos utilizando muestras aleatorias con reemplazo, y luego combinar los resultados. De esta forma, podemos ver que algunos errores se compensan obteniendo asi una mejor prediccion. Cuanto mayor cantidad de arboles, mejor sera la prediccion.
A su vez, podemos notar que los hiperparametros mas utiles para este modelo predictivo son:

- **n_estimators**: es el numero de arboles que utilizaremos para predecir. Al utilizar múltiples árboles reducimos considerablemente el riesgo de overfitting. Esto tiene una desventaja: a partir de cierta cantidad, deja de ser buen predictor y lo único que genera es mayor costo computacional.
- **max_features**: manera de seleccionar la cantidad maxima de features para cada arbol.
- **max_depth**: profundidad maxima del arbol.
- **min_sample_leaf**: numero minimo de elementos en las hojas.
- **boostrap**: para utilizar diferentes tamaños de muestras. Si se pone en False, utiliza el dataset completo.

Los modelos ensambladores de arboles, como en el caso del Random Forest, sufren problemas de sesgo (promedio entre los valores predecidos y los valores reales) y varianza

(cuán diferentes son las predicciones teniendo en cuentas diferentes muestras tomadas de la misma poblacion). Un modelo optimo, buen predictor, sera aquel que mantenga un balance entre estos dos problemas. Algunas de las desventajas mas importantes de este modelo son las siguientes:

1. No esta creado para trabajar con datasets pequeños.
2. Puede requerir muchisimo tiempo de entrenamiento.
3. Es mucho mas costoso si la cantidad de arboles es demasiado alta.

3. Conclusiones

Durante la realización de este trabajo aplicamos una ingeniería de características que mejoraron las predicciones notablemente, esto nos demuestra la importancia de trabajar la información con criterios teóricos aplicados en el entrenamiento del modelo.

También concluímos que a pesar de que el clasificador KNN demuestra buenos resultados, el modelo más óptimo es el XGBoost ya que presentó mejor puntiuación.

4. Referencias

- `https://www.drivendata.org/competitions/57/nepal-earthquake/page/134/`
- `https://github.com/Cbravor1991/ModeloMachineLearning`