

7506-2021 TP 2

Román Vázquez Lareu, Matías Ignacio González

TOTAL POINTS

68 / 100

QUESTION 1

1 Informe 30 / 40

- **0 pts** Correct

✓ - **10 pts** Falta profundidad en la explicación del trabajo de preprocesamiento, feature engineering, etc

- **10 pts** Falta el detalle de las pruebas realizadas, solo incluyen comentarios muy generales sobre el trabajo

- **10 pts** Fantan conclusiones

- **30 pts** Trabajo muy incompleto. Falta desarrollo

- **5 pts** Falta profundidad en las conclusiones

QUESTION 2

2 Competencia 24 / 40

- **0 pts** Correct

- **16 Point adjustment**

💬 0.7446

QUESTION 3

3 Video Presentación 14 / 20

- **0 pts** Correct

- **20 pts** Falta link al video

- **20 pts** El link provisto esta inaccesible

- **6 Point adjustment**

💬 La presentacion esta bien, podria tener mas detalles pero esta bien hilada, tiene sentido y participan ambos integrantes.



Trabajo Práctico 2

[7506] Organización de Datos

Curso Argerich

Primer cuatrimestre de 2021

VAZQUEZ LAREU, Román	100815
GONZALEZ, Matías Ignacio	104913

Índice

1. Introducción	2
2. Experimentos	2
2.1. Información previa a realizar el experimento a tener en cuenta	2
2.2. Experimento 1	2
2.3. Experimento 2	3
2.4. Información previa a realizar el experimento a tener en cuenta	3
2.5. Experimento 3	3
2.6. Experimento 4	4
2.7. Experimento 5	4
2.8. Información previa a realizar el experimento a tener en cuenta	5
2.9. Experimento 6	5
2.10. Experimento 7	6
2.11. Información previa a realizar el experimento a tener en cuenta	6
2.12. Experimento 8	8
2.13. Experimento 9	8
2.14. Información previa a realizar el experimento a tener en cuenta	9
2.15. Experimento 10	9
2.16. Información previa a realizar el experimento a tener en cuenta	10
2.17. Experimento 11	10
2.18. Experimento 12	10
2.19. Experimento 13	11
2.20. Experimento 14	12
2.21. Experimento 15	12
2.22. Experimento 16	12
2.23. Experimento 17	13
2.24. Experimento 18	13
3. Resultado en Driven Data	14
4. Conclusiones	14
5. Bibliografía	15
6. Links	15

1. Introducción

El presente informe reune la documentación de la realización del segundo trabajo práctico de la materia Organización de Datos que consiste en predecir el nivel de daño a los edificios causado por el terremoto Gorkha en Nepal en el 2015, basándose en distintos aspectos de la construcción y su localización.

La realización se presentará como la secuencia de experimentos realizados, indicando su motivación, modelo utilizado, Features tenidos en cuenta en el entrenamiento del modelo, grilla de parámetros para hacer tuning, los mejores obtenidos luego de la búsqueda, el puntaje de f1 obtenido, el correspondiente puntaje obtenido en la competencia en Driven Data y finalmente la conclusión u observación extraída del experimento.

2. Experimentos

2.1. Información previa a realizar el experimento a tener en cuenta

Random forest se comporta bien con la mayoría de los set de datos. Cada árbol usa un subconjunto de atributos y usa un bootstrap del set de entrenamiento. El conjunto de atributos tomados es al azar.

En cuanto a los hiperparametros los mas destacables son *n_estimators* que es la cantidad de árboles (cuántos más mejor pero baja la performance) y la cantidad de atributos por arbol. Una instancia de GridSearchCV implementa la API de un estimador: cuando se entrena con un dataset, se evaluan todas las posibles combinaciones y se retiene la mejor

2.2. Experimento 1

Motivación Punto de partida sobre el cual comenzar a trabajar

Modelo Random Forest

Features Usados Todos salvo categóricos y *secondary_use*

Grilla de parámetros y otros

$$n_estimators = [50, 100]$$

$$min_samples_leaf = [1, 5]$$

$$GridSearch_cv = 5$$

Mejores Parámetros

$$n_estimators = [100]$$

$$min_samples_leaf = [5]$$

f1 score $f1_score = 0,7713$

Driven Data Score $Driven_Data_Score = 0,7283$

Conclusión/observaciones Posible overfitting, igualmente buen punto de partida en comparación con el resto de los puntajes y el benchmark provisto por la plataforma.

2.3. Experimento 2

Motivación Evidenciar el posible overfitting, teniendo en cuenta todas las columnas y haciendo OneHotEncoding de todos los features categóricos. Así como el experimento anterior funcionará de benchmark, este lo hará para el overfitting.

Modelo Random forest

Features Usados Todos los features, sumadas las columnas generadas por *oneHotEncoding* de los categóricos

Grilla de parámetros y otros

$$n_estimators = [50, 100]$$

$$min_samples_leaf = [1, 5]$$

$$GridSearch_cv = 5$$

Mejores Parámetros

$$n_estimators = [100]$$

$$min_samples_leaf = [5]$$

f1 score $f1_score = 0,9807$

Driven Data Score $Driven_Data_Score = 0,7260$

Conclusión/observaciones El resultado era el esperado. El modelo generaliza mal, alucina. Se evidencia el *overfitting*

2.4. Información previa a realizar el experimento a tener en cuenta

A la búsqueda de hiper-parámetros con el objetivo de encontrar un set óptimo que minimice una función de pérdida para dar mejores resultados se lo llama *Tunning de Hiper-parámetros*

2.5. Experimento 3

Motivación Corregir resultado anterior mediante la búsqueda de hiper-parámetros que mejoren el resultado.

Modelo Random forest

Features Usados Igual a experimento anterior

Grilla de parámetros y otros

$$n_estimators = [50, 150]$$

$$min_samples_leaf = [1, 5]$$

$$GridSearch_cv = 5$$

Mejores Parámetros

$$n_estimators = [150]$$

$$min_samples_leaf = [5]$$

f1 score $f1_score = 0,7540$

Driven Data Score $Driven_Data_Score = 0,7108$

Conclusión/observaciones Efectivamente el modelo con todos los atributos y más tenidos en cuenta al momento de entrenar generaliza mal, sin embargo el uso de en este caso más arboles mejoró el resultado. Podría hacerlo aún más ya que el parámetro se encontraba en el límite superior de la grilla.

2.6. Experimento 4

Motivación Corregir resultado anterior mediante la búsqueda de hiper-parámetros que mejoren el resultado.

Modelo Random forest

Features Usados Igual a experimento anterior

Grilla de parámetros y otros

$$n_estimators = [150, 200]$$

$$min_samples_leaf = [1, 5]$$

$$GridSearch_cv = 5$$

Mejores Parámetros

$$n_estimators = [200]$$

$$min_samples_leaf = [5]$$

f1 score $f1_score = 0,7431$

Driven Data Score $Driven_Data_Score = 0,7267$

Conclusión/observaciones Mejora respecto al anterior, habría que empezar a ver resultados modificando features

2.7. Experimento 5

Motivación Mejorar el resultado obtenido haciendo feature engineering

Modelo Random forest

Features Usados Se agregan interacciones entre algunos features que resultaron de importancia:

$$floor_percentage/height_percentage$$

$$floor_percentage/area_percentage$$

Grilla de parámetros y otros

$$n_estimators = [150, 200]$$

$$min_samples_leaf = [1, 5]$$

$$GridSearch_cv = 5$$

Mejores Parámetros

$$n_estimators = [200]$$

$$min_samples_leaf = [5]$$

f1 score $f1_score = 0,7431$

Driven Data Score $Driven_Data_Score = 0,7033$

Conclusión/observaciones Se esperaba una mejora pero se obtuvo el peor resultado. El ratio entre los atributos de importancia debería influir positivamente en el modelo y generalmente lo hacen. El problema podría estar en el modelo mismo o en los hiper-parámetros.

2.8. Información previa a realizar el experimento a tener en cuenta

XGBoost consiste en un boosting de árboles de decisión, donde cada árbol corrige lo que hizo mal el anterior.

HalvingGridSearchCV es capaz de encontrar combinaciones de parámetros que son tan precisas como *GridSearchCV* en mucho menos tiempo. Esto lo hace de la siguiente manera: comienza evaluando todos los candidatos con una cantidad limitada de recursos y en las sucesivas iteraciones conserva los mejores y va usando más recursos para evaluarlos.

2.9. Experimento 6

Motivación Mejorar el resultado obtenido con random forest, utilizando XGBoost. Ver como se comporta en comparación con random Forest

Modelo XGBoost

Features Usados Igual a experimento anterior

Grilla de parámetros y otros

```
base_score : [0,5]
colsample_bytree : [0,4,0,6]
learning_rate : [0,1,0,2]
max_delta_step : [0]
max_depth : [4,7,9]
n_estimators : [100,200]
subsample : [0,7,0,8]
HalvingGridSearchcv = 3
```

Mejores Parámetros

```
base_score : [0,5]
colsample_bytree : [0,6]
learning_rate : [0,1]
max_delta_step : [0]
max_depth : [9]
n_estimators : [100]
subsample : [0,8]
```

f1 score $f1_score = 0,7281$

Driven Data Score $Driven_Data_Score = 0,7267$

Conclusión/observaciones Iguala el mejor resultado junto con random Forest, a su vez el f1 da similar al resultado obtenido en driven Data. Indicaría que el modelo se encuentra en un buen lugar entre underfitting y overfitting. Vale la pena intentar mejorarlo ya sea con hiper-parámetros o con feature engineering

2.10. Experimento 7

Motivación Mejorar el resultado anterior con hiper-parámetros, probando con $cv = 5$ en vez de 3 y mediante OOB

Modelo XGBoost

Features Usados Igual a experimento anterior

Grilla de parámetros y otros

```
base_score : [0,5]
colsample_bytree : [0,6,0,7]
learning_rate : [0,05,0,1]
max_delta_step : [0]
max_depth : [9,13]
n_estimators : [100,200]
subsample : [0,8,0,9]
```

HalvingGridSearchcv = 5

Mejores Parámetros

```
base_score : [0,5]
colsample_bytree : [0,6]
learning_rate : [0,1]
max_delta_step : [0]
max_depth : [13]
n_estimators : [100]
subsample : [0,8]
```

f1 score $f1_score = 0,7408$

Driven Data Score $Driven_Data_Score = 0,7396$

Conclusión/observaciones La mejora fue mayor a la esperada. Por el momento se prioriza trabajar con XGBoost a hacerlo con random Forest. sin embargo la diferencia entre $f1_score$ y $Driven_Data_Score$ fue mayor al caso anterior. ¿Posible overfitting?

2.11. Información previa a realizar el experimento a tener en cuenta

Con el objetivo de reducir el posible overfitting se tendrá en cuenta la siguiente características del dataset, que muestran que el porcentaje de edificios con uso secundario es bajo dentro set de datos y que se encuentra desigualmente distribuido entre los diferentes tipos de uso

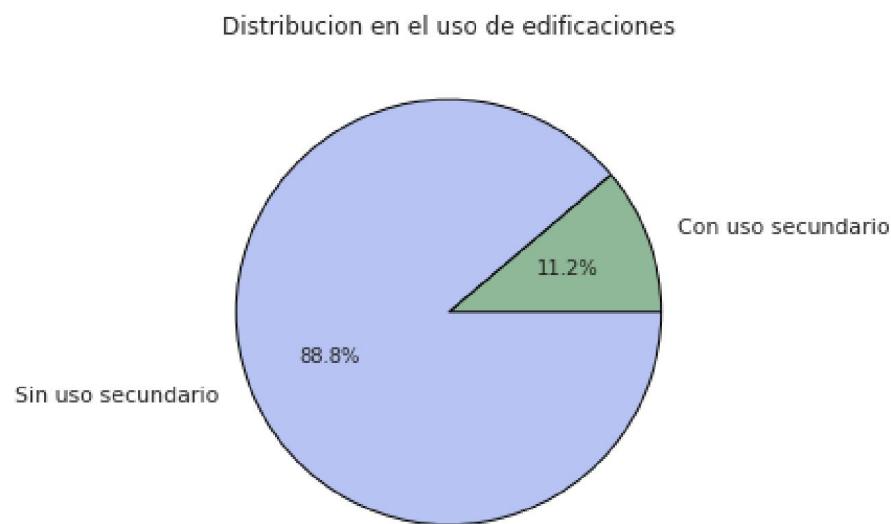


Figura 1: Solo el 11,2 % del dataset se corresponde a edificaciones con uso secundario

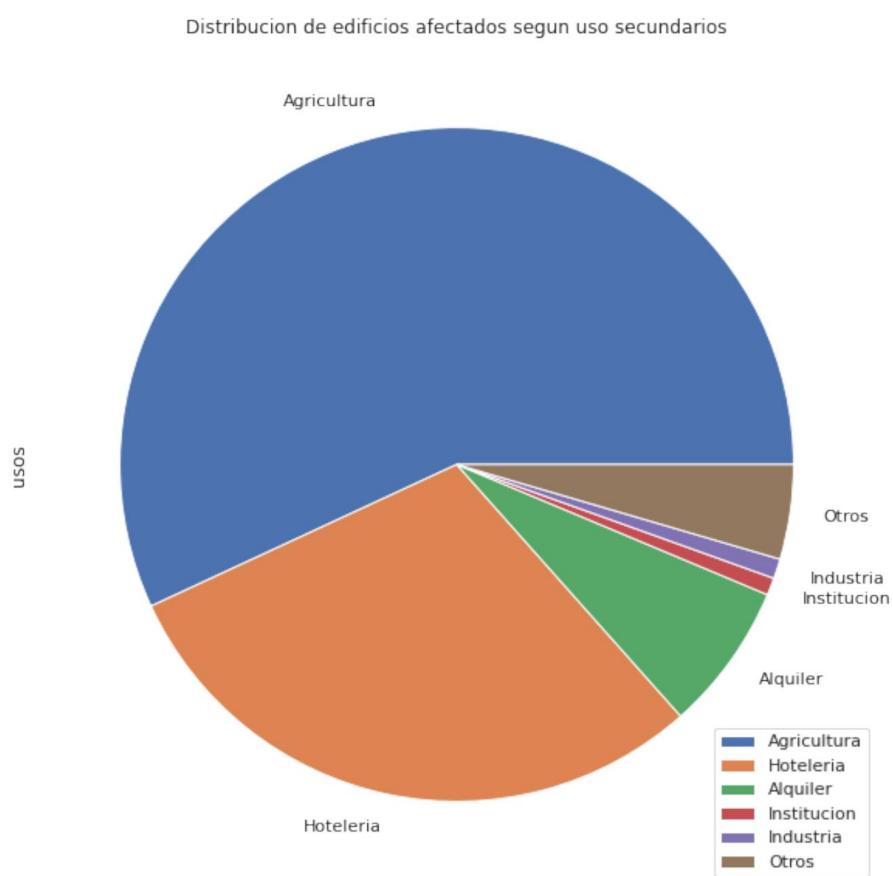


Figura 2: Del porcentaje de edificaciones con uso secundario, la mayoria se reparte entre hoteleria y agricultura

2.12. Experimento 8

Motivación Corregir el posible overfitting haciendo un merge entre columnas cuya participación en el dataset sea baja, en este caso la de usos secundarios

Modelo XGBoost

Features Usados Se agrega un merge entre *secondary_uses* que tienen poca participación dentro del dataset mediante un *or* entre Alquiler, Institución, Industria y otros, reduciendo este conjunto de columnas a 4: no tiene uso secundario, tiene uso hotelero, tiene uso agricultura, tiene uso otro (engloba a la mayoría)

Grilla de parámetros y otros Se mantuvieron los parametros anteriores

Mejores Parámetros Se mantuvieron los parametros anteriores

f1 score $f1_score = 0,7421$

Driven Data Score $Driven_Data_Score = 0,7417$

Conclusión/observaciones Mejoro el resultado en driven data y se achicó la brecha entre el f1 local y este resultado. Era lo esperado. A su vez se mejoró el resultado global

2.13. Experimento 9

Motivación Mejorar haciendo feature engineering

Modelo XGBoost

Features Usados Se agrega una columna *height_percentage/area_percentage*, ya que presentó una fuerte correlacion con los niveles de daño.

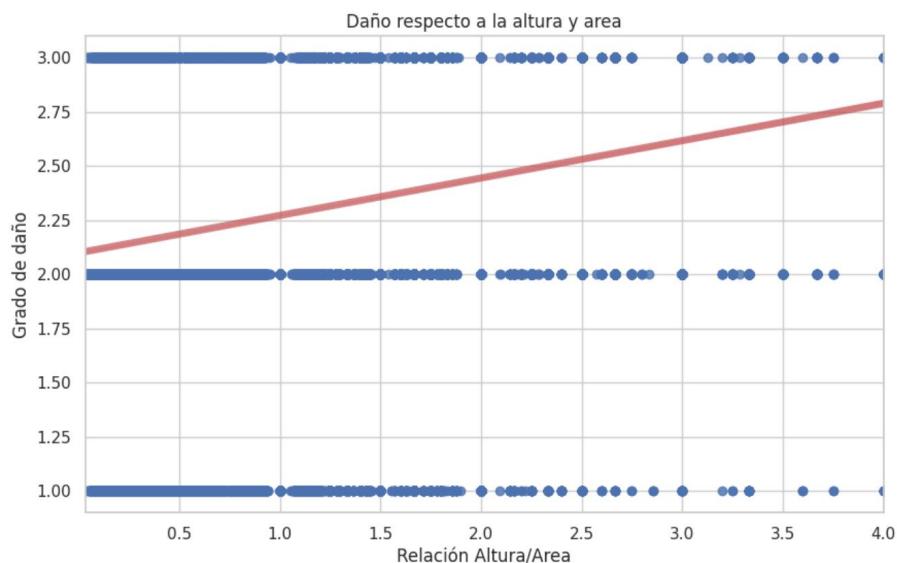


Figura 3: Correlación Altura y Area

Grilla de parámetros y otros Se mantuvieron los parametros anteriores

Mejores Parámetros Se mantuvieron los parametros anteriores

f1 score $f1_score = 0,7432$

Driven Data Score $Driven_Data_Score = 0,7411$

Conclusión/observaciones El resultado fue peor, aunque por muy poco. El error debe venir de otro lado, ya que el ratio $height_percentage/area_percentage$ se da entre los features de mayor importancia luego de geo_level_id y debería influir positivamente. Podría ser que los hiper-parámetros no sean los óptimos (dentro de los tenidos en cuenta) para este set.

2.14. Información previa a realizar el experimento a tener en cuenta

El parámetro *objective* es un parámetro de *learning*. En los casos anteriores se venía utilizando el *binary : logistic*, el cual era una regresión logística para una clasificación binaria con una probabilidad como output.

multi : softprob es más adecuado, ya que prepara a XGBoost para hacer una clasificación multiclasa usando la función softmax. La salida es un vector de $N_{data} * N_{class}$. El resultado contiene la probabilidad predecida para cada punto de pertenecer a cada clase.

2.15. Experimento 10

Motivación Corregir lo anterior con la búsqueda de hiper-parametros y modificando el *objective*

Modelo XGBoost

Features Usados Igual a experimento anterior

Grilla de parámetros y otros Se modificó el *objective* a "multisoft:prob"

```
base_score : [0,5]
colsample_bytree : [0,6,0,7]
learning_rate : [0,05,0,1]
max_delta_step : [0]
max_depth : [9,13]
n_estimators : [100,120]
subsample : [0,8,0,9]
HalvingGridSearchcv = 5
```

Mejores Parámetros

```
base_score : [0,5]
colsample_bytree : [0,6]
learning_rate : [0,1]
max_delta_step : [0]
max_depth : [13]
n_estimators : [120]
subsample : [0,8]
HalvingGridSearchcv = 5
```

f1 score $f1_score = 0,7443$

Driven Data Score $Driven_Data_Score = 0,7416$

Conclusión/observaciones Mejora el resultado a prácticamente el anterior mejor. Vale la pena tener en cuenta $height_percentage/area_percentage$. Se buscará mejorar de otra manera. Ya sea con Feature engineering o haciendo una limpieza del dataset (convendría haberlo hecho desde el principio)

2.16. Información previa a realizar el experimento a tener en cuenta

Los Outliers son los puntos en el dataset que difieren significativamente del resto de las observaciones. Puede ocurrir tanto por una variación en el mismo como por un error. La búsqueda que hicimos fue de outliers de una dimensión, podría haberse hecho de más también. Un método puede ser el *Z_score*, el cual asume que la variable tiene una distribución gaussiana. Representa el numero de desviaciones estandar a las que se encuentra una observacion de la media. Los outliers se definen como aquellos cuyo módulo de *Z_score* es mayor a un cierto numero de desviaciones estandar (2 o 3 generalmente).

Otro método con el objetivo de minimizar la influencia de outliers es el de *winsorizing*, difiere del anterior en que los outliers son modificados para no afectar de gran manera el dataset y no eliminados. Para eso se debe establecer un límite superior e inferior del set de datos a *winsorize*.

2.17. Experimento 11

Motivación Empujar el dataset y ver si hay leve mejora

Modelo XGBoost

Features Usados Igual a experimento anterior pero se eliminan los outliers de *age*, *height_percentage* y *floor_percentage* con *winsorize*

Grilla de parámetros y otros Igual a experimento anterior

Mejores Parámetros Igual a experimento anterior Se mantuvieron los parametros anteriores

f1 score *f1_score* = 0,7439

Driven Data Score *Driven_Data_Score* = 0,7421

Conclusión/observaciones Mejora el resultado que era lo esperado, evidencia que los outliers perjudican el rendimiento de los árboles de decisión.

2.18. Experimento 12

Motivación La feature *geo_level_id* es numérica pero no tiene un orden que tenga sentido para el árbol de decisiones. Lo que quiere decir es que se está imponiendo un orden totalmente arbitrario. Por esta razón es necesario encodear estas features de otra manera.

Modelo XGBoost

Features Usados Todos los features salvo secondary uses, sumadas las columnas generadas por oneHotEncoding de los categoricos menos las *geo_level_id*

Grilla de parámetros y otros

```

base_score : [0,5]
colsample_bytree : [0,6]
learning_rate : [0,1]
max_delta_step : [1]
max_depth : [13, 20, 25]
n_estimators : [120]
subsample : [0,8]
HalvingGridSearchcv = 5

```

Mejores Parámetros

max_depth : 13

f1 score *f1_score = 0,617*

Driven Data Score No se hizo un submit de esta etapa.

Conclusión/observaciones A pesar de estar muy mal encodeado, esta feature ayuda a la predicción de los datos, y mejora el score. De todas maneras se debe encodear de otra manera.

2.19. Experimento 13

Motivación Se vio durante el análisis exploratorio de datos que las features Age y cantidad de pisos podrían estar relacionadas. No se pudo encontrar una clara correlación pero se decide intentar incluir la feature Age*Floors.

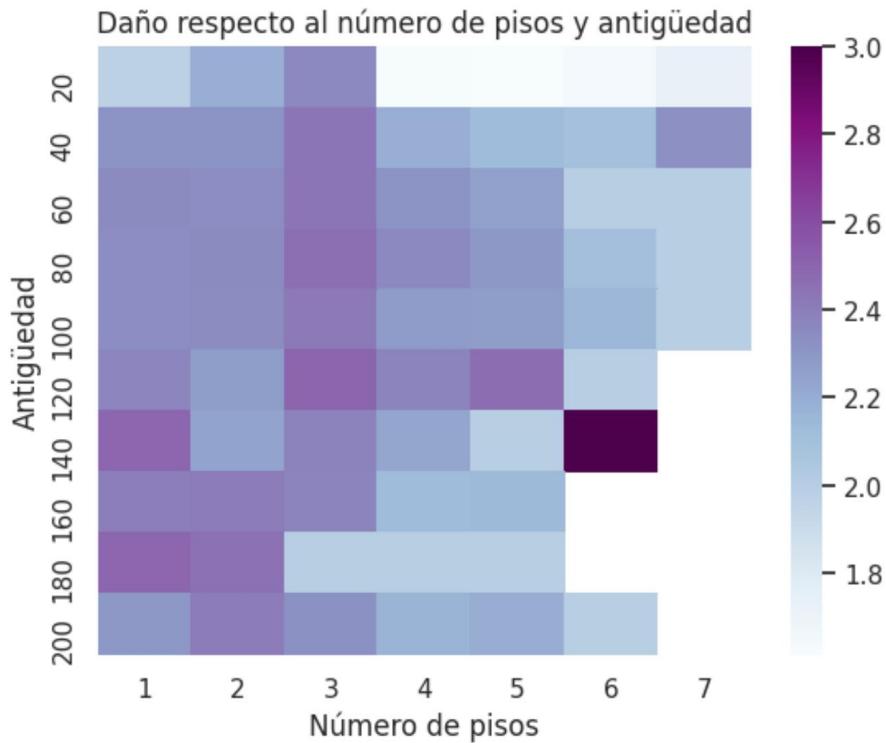


Figura 4: Heatmap Piso y Age

Modelo XGBoost

Features Usados Todos los features anteriores + Age*Floors *geo_level_id*

Grilla de parámetros y otros Se usaron los parámetros del experimento anterior.

f1 score *f1_score = 0,744*

Driven Data Score *Driven_Data_Score = 0,7411*

Conclusión/observaciones El Score empeoro con respecto a experimentos anteriores. Se decide no usar esta nueva feature.

2.20. Experimento 14

Motivación Mejorar los resultados obtenidos con feature engineering y buscando reducir el overfitting. Hasta ahora siempre el f1 fue mayor al resultado de drivenData

Modelo XGBoost

Features Usados Se eliminan las columnas de *geo_level_id* y se las reemplaza por una única columna. A partir de entrenar un XGBoost únicamente basado en los *geo_level_id*, se hace una predicción y ese valor es el que reemplaza a los *geo_level_id*. Se espera que mejore y reduzca overfitting

Grilla de parámetros y otros Igual a experimento 11

Mejores Parámetros Igual a experimento 11

f1 score $f1_score = 0,7376$

Driven Data Score $Driven_Data_Score = 0,7234$

Conclusión/observaciones Mal resultado. Evidentemente el algoritmo planteado no reemplaza a los *geo_level_id* o sus hiperparámetros no fueron correctos. También se debe tener en cuenta los hiper-parámetros y feature engineering del árbol entrenado con los 3 *geo_level_id*

2.21. Experimento 15

Motivación Similar al caso anterior pero con las columnas de *secondary_use*

Modelo XGBoost

Features Usados Se eliminan las columnas de *secondary_use* y se las reemplaza por una única columna. A partir de entrenar un XGBoost únicamente basado en los *secondary_use*, se hace una predicción y ese valor es el que reemplaza a los *secondary_use*. Se espera que mejore y reduzca overfitting

Grilla de parámetros y otros Igual a experimento anterior

Mejores Parámetros Igual a experimento anterior

f1 score $f1_score = 0,7276$

Driven Data Score $Driven_Data_Score = 0,7252$

Conclusión/observaciones Mejor respecto al anterior pero malo en general. Evidencia que no es reemplazable por una columna de este tipo o que fallaron los hiper-parámetros del algoritmo usado en el XGBoost del árbol entrenado con *secondary_use*

2.22. Experimento 16

Motivación Se descartan los dos atributos previos y se vuelve al mejor resultado obtenido (experimento 9). Se buscará mejorarlo buscando mejores hiper-parámetros

Modelo XGBoost

Features Usados Los correspondientes al experimento 9.

Grilla de parámetros y otros

$n_estimators = [120, 150]$

Mejores Parámetros

$$n_estimators = [150]$$

f1 score $f1_score = 0,7463$

Driven Data Score $Driven_Data_Score = 0,7426$

Conclusión/observaciones Mejora leve debido a hallar mejores hiperparametros. Era lo esperado, aunque a su vez pone tope al rendimiento con este conjunto de datos. O hacer feature Engineering o invertir mucho tiempo en *Tunning* de hiperparámetros

2.23. Experimento 17

Motivación Agregar a un modelo con los mejores hiper-parámetros conseguidos (el anterior), la columna obtenida en el experimento 13 (predicciones obtenidas a partir de entrenar un XGBoost con los *geo_level_id*. En este caso sin eliminar ninguno de los 3 *geo_level_id*. Podría mejorar o empeorar por overfitting

Modelo XGBoost

Features Usados Se agrega la columna con las predicciones correspondientes al modelo entrenado con los *geo_level_id*

Grilla de parámetros y otros igual a experimento anterior

Mejores Parámetros igual a experimento anterior

f1 score $f1_score = 0,7530$

Driven Data Score $Driven_Data_Score = 0,7446$

Conclusión/observaciones Mejora notable en el resultado, evidencia que la columna suma al modelo y es influyente en la toma de decisiones del mismo. Posible overfitting

2.24. Experimento 18

Motivación Teniendo en cuenta el resultado anterior, se hará lo mismo pero con las predicciones en base al modelo entrenado con los *secondary_uses*

Modelo XGBoost

Features Usados Se agrega la columna con las predicciones correspondientes al modelo entrenado con los *secondary_uses*

Grilla de parámetros y otros igual a experimento anterior

Mejores Parámetros igual a experimento anterior

f1 score $f1_score = 0,7474$

Driven Data Score $Driven_Data_Score = 0,7432$

Conclusión/observaciones No mejora el resultado. Las predicciones del arbol de los usos tenían un f1 de solo 0.56. Puede que no haya sido suficiente para influir positivamente en el modelo. También puede ser que sean necesarios otros hiper-parámetros o que haya overfitting.

3. Resultado en Driven Data

El mejor Score en Driven Data fue obtenido luego del experimento 17.



Figura 5: Ranking en Driven Data

4. Conclusiones

Una vez finalizado el trabajo y con la experiencia ya adquirida, la metodología para realizarlo nuevamente hubiera sido distinta. Idealmente hubiese sido planteando prototipos para varios modelos (random forest, XGBoost, Light-GBM por ejemplo) e ir realizando feature engineering en paralelo, junto con tuning de hiper-parámetros. En nuestro caso se hizo de manera secuencial y cuando se detectó que para un cierto set de datos XGBoost daba mejores resultados que Random Forest se descartó este último y se continuó únicamente con el modelo XGBoost.

En cuanto al modelo final, la búsqueda de hiper-parámetros por grilla o fuerza bruta dio buenos resultados dado que el set de datos no era de gran tamaño.

En la etapa de Feature Engineering, el hecho de agregar features basados en árboles de decisión externos al modelo principal dio buenos resultados. Si bien oneHotEncoding no es recomendable ya que puede agregar muchas dimensiones, en este caso al ser pocos los distintos valores por atributo categórico, influyó de manera positiva en el modelo. Los *geo_level_id* son los que influyen a la hora de tomar decisiones en el árbol, a pesar de esto, al ser numéricos se les impone un orden arbitrario, por lo que podría haber sido una buena idea encodearlos de tal manera de romper con ese orden preestablecido sin sentido.

Otro elemento a destacar como positivo fue la interacción (especialmente ratio) entre features de importancia para el modelo o con una fuerte correlación con el *damage_grade*. Finalmente, ayudó en la reducción del overfitting y en obtener una mejor generalización la reducción de columnas correspondientes a los usos secundarios, en este caso agrupando las que tenían poca participación dentro del dataset.

Para finalizar, en lo que corresponde con la elección al modelo, como se mencionó anteriormente hubiese sido preferible llevar varios en paralelo y no secuencialmente ir de uno a otro. A pesar de esto, fue de utilidad partir de un modelo random Forest ya que tiene buenos resultados para la mayoría de los set de datos y sirvió como punto de partida para desarrollar el trabajo práctico.

5. Bibliografía

- Material de la cátedra
- <https://scikit-learn.org/>
- <https://xgboost.readthedocs.io/>

6. Links

- [Video](#)
- [Repositorio](#)
- [Colab](#)

1 Informe 30 / 40

- **0 pts** Correct
- ✓ - **10 pts** Falta profundidad en la explicación del trabajo de preprocesamiento, feature engineering, etc
- **10 pts** Falta el detalle de las pruebas realizadas, solo incluyen comentarios muy generales sobre el trabajo
- **10 pts** Fantan conclusiones
- **30 pts** Trabajo muy incompleto. Falta desarrollo
- **5 pts** Falta profundidad en las conclusiones

3. Resultado en Driven Data

El mejor Score en Driven Data fue obtenido luego del experimento 17.



Figura 5: Ranking en Driven Data

4. Conclusiones

Una vez finalizado el trabajo y con la experiencia ya adquirida, la metodología para realizarlo nuevamente hubiera sido distinta. Idealmente hubiese sido planteando prototipos para varios modelos (random forest, XGBoost, Light-GBM por ejemplo) e ir realizando feature engineering en paralelo, junto con tuning de hiper-parámetros. En nuestro caso se hizo de manera secuencial y cuando se detectó que para un cierto set de datos XGBoost daba mejores resultados que Random Forest se descartó este último y se continuó únicamente con el modelo XGBoost.

En cuanto al modelo final, la búsqueda de hiper-parámetros por grilla o fuerza bruta dio buenos resultados dado que el set de datos no era de gran tamaño.

En la etapa de Feature Engineering, el hecho de agregar features basados en árboles de decisión externos al modelo principal dio buenos resultados. Si bien oneHotEncoding no es recomendable ya que puede agregar muchas dimensiones, en este caso al ser pocos los distintos valores por atributo categórico, influyó de manera positiva en el modelo. Los *geo_level_id* son los que influyen a la hora de tomar decisiones en el árbol, a pesar de esto, al ser numéricos se les impone un orden arbitrario, por lo que podría haber sido una buena idea encodearlos de tal manera de romper con ese orden preestablecido sin sentido.

Otro elemento a destacar como positivo fue la interacción (especialmente ratio) entre features de importancia para el modelo o con una fuerte correlación con el *damage_grade*. Finalmente, ayudó en la reducción del overfitting y en obtener una mejor generalización la reducción de columnas correspondientes a los usos secundarios, en este caso agrupando las que tenían poca participación dentro del dataset.

Para finalizar, en lo que corresponde con la elección al modelo, como se mencionó anteriormente hubiese sido preferible llevar varios en paralelo y no secuencialmente ir de uno a otro. A pesar de esto, fue de utilidad partir de un modelo random Forest ya que tiene buenos resultados para la mayoría de los set de datos y sirvió como punto de partida para desarrollar el trabajo práctico.

2 Competencia 24 / 40

- 0 pts Correct

- 16 Point adjustment

 0.7446

5. Bibliografía

- Material de la cátedra
- <https://scikit-learn.org/>
- <https://xgboost.readthedocs.io/>

6. Links

- [Video](#)
- [Repositorio](#)
- [Colab](#)

3 Video Presentación 14 / 20

- **0 pts** Correct
 - **20 pts** Falta link al video
 - **20 pts** El link provisto esta inaccesible
- 6 Point adjustment**

 La presentacion esta bien, podria tener mas detalles pero esta bien hilada, tiene sentido y participan ambos integrantes.