UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA

75.06 Organización de Datos
Trabajo Práctico N°2
Primer Cuatrimestre de 2021

Grupo: Rasdatos

Integrantes: Spiguelman, Nahuel Hernan
Puquio, Giancarlo José
Gonzalez, Francisco Javier

Objetivo:

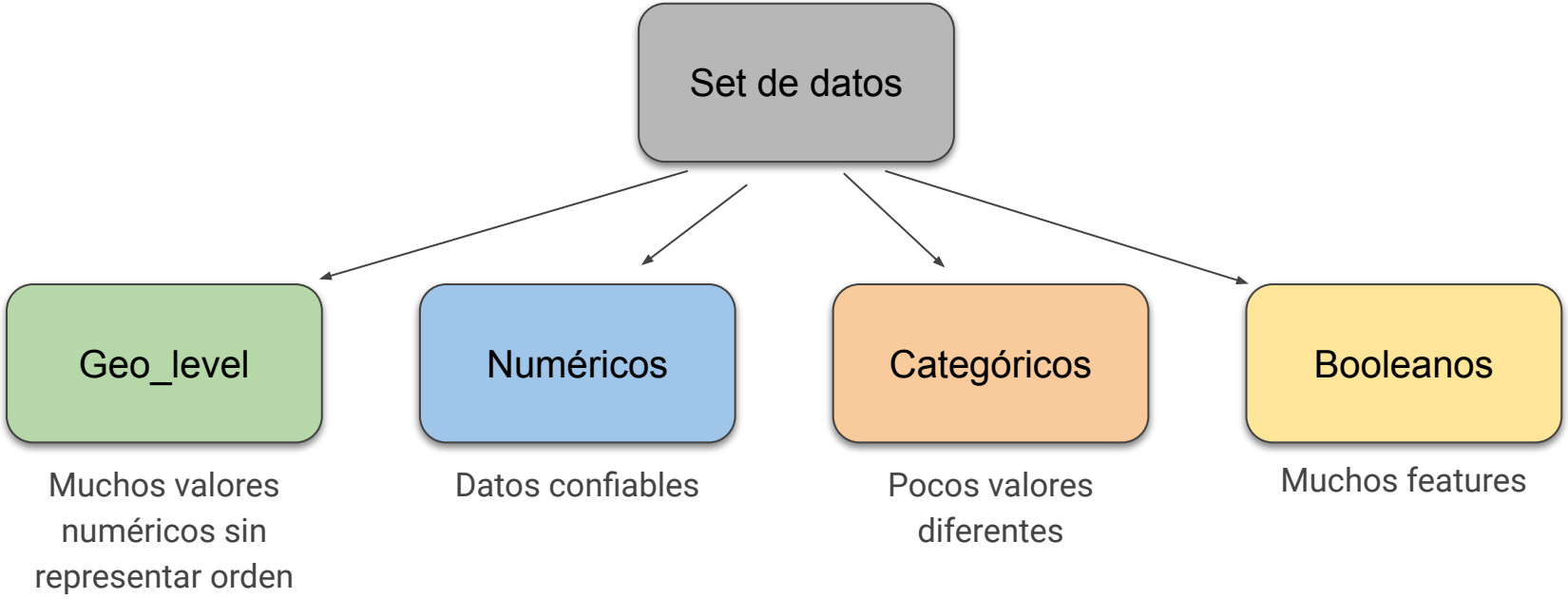
Predecir el nivel de daño representado por la variable ‘damage_grade’ (métrica: ‘f1_micro’)

El TP2 es continuación del TP1 donde realizamos un análisis exploratorio de los datos.
A partir de lo observado en el TP1, los invitamos a conocer el camino recorrido en este TP2 ---->

Richter's Predictor: Modeling Earthquake Damage

Hosted By DrivenData

Features



Premisa:

Conservar la mayor cantidad de features posibles, priorizando aquellos más relevantes del TP1

Búsqueda de Hiper-parámetros

Etapas:

- Búsqueda artesanal
- Grid Search
- Random Search
- Early Stopping

Random Forest

Primer modelo de clasificación
- Reducción de features
- Sin busqueda de HP

Test: 0.58
Submit: 0.5780

Feature Engineering
- Categóricos
- OneHot Encoding

Test: 0.6296
Submit: 0.6134

Optimización de HP
- Grid Search
- Random Search

Test: 0.6919
Submit: 0.6953

Conclusiones:
Fue el primer modelo que probamos, y nos ayudó a comprender cómo debíamos trabajar y empezar a incorporar herramientas que nos resultaban desconocidas.

Test: 0.6919
Submit: 0.6953

KNN

Primer modelo (15 features)

- Top 15 features más relevantes según R.Forest
- K=28. K óptimo calculado con el promedio del error (clasificaciones incorrectas)

Test: 0.6536
Submit: -

Feature engineering

- Prueba de diferentes modelos cambiando features y reduciendo la de cantidad de los mismos
- Mejor modelo:
  3 features geo_level
  K= 18

Test: 0.7087
Submit: 0.4593

Feature engineering
Reducción de dimensiones - PCA

- Reducción de 15 a 10 features estimados con el cálculo de la varianza explicada
- K= 10

Test: 0.6681
Submit: 0.5812

Conclusiones:
Experimentamos un claro overfiting motivados por el resultado, y obtuvimos mejores resultdos a concentrarnos nuevamente en los datos y utilizar redución de dimensiones de los featrures más importantes

XGBoost

- XGBoost Classifier
- Random Search

Hiper-parámetros

subsample: 0.8
num_parallel_tree: 2
n_estimators: 500
max_depth: 12
learning_rate: 0.1
colsample_bytree: 0.5

Feature engineering

Se realizó un encoding de las variables categóricas del dataset con one hot encoding.

Conclusiones:
Mejores resultados en comparación con Random Forest.

Tiempo para la búsqueda de hiperparametros fue significativamente grande.

Test: 0.7023
Submit: 0.6981

Logistic Regression

Logistic Regression:
- Feature Engineering
- sin búsqueda de HP

Hiper-parámetros
- Random Search
  - C: 20
  - penalty: ‘l2’

Train: 0.66
Test: 0.65

Conclusiones:
No se obtuvieron buenos resultados, por lo cual decidimos seguir probando con modelos basados en árboles de decisiones.

Gradient Boosting

- Gradient Boosting Classifie
- Random Search

Hiper-parámetros

max_depth: 8
subsample: 1
n_estimators: 500
learning_rate: 0.1

Feature engineering

Se utilizó el mismo feature engineering utilizado en Xgboost

Conclusiones:

No se obtuvieron mejores resultados que otros modelos.

Test: 0.6980
Submit: -

CatBoost

Primer modelo
- Hiper-parámetros por defecto
- Data set sin Geo level 2 y 3 (Sesgo)

Test: 0.6938
Submit: 0.6919

Hiper-parámetros
- RandomSearchCV - Busqueda manual
- Mucho tiempo - poca mejora

Test: 0.6945
Submit: 0.6936

Feature engineering
- Agregamos Geo_level 2 y 3 (Mejora importante)

Test: 0.7486
Submit: 0.7472

Mejor modelo
- Agregamos 2 features “has_
- Continuamos busqueda de hiper_parámetros

Test: 0.7505
Submit: 0.7498

Conclusiones:
En este modelo fue crucial el feature engineering. El principal salto lo dimos al incluir las categorías Geo_level 2 y 3, sesgo que arrastrabamos de los modelos anteriores. También sumo la creación de los features #has_#

Modo competencia (best model - full train) → Submit: 0.7533