GRADIENT BOOSTING

Y aplicaciones

Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Victor Manuel Gómez Espinosa

1. INTRODUCCIÓN

- Gradient Boosting Friedman (2001)
- originalmente para el caso de regresión y que posteriormente se extendió para el caso de clasificación
- la implementación original de este algoritmo fue llamada MART "*Multiple additive regression trees*"

1. INTRODUCCIÓN

- XGBoost (Chen and Guestrin 2016)
- la implementación de aprendizaje maquina más utilizada (seguida de Deep neural networks) por la mayoría de los equipos ganadores en diversos concursos (Netflix prize, Kaggle, KDDCup)
- Algunos de los problemas de estas competencias son: predicción de ventas, predicción del comportamiento de clientes, clasificación de textos, predicción de riesgos, entre otros.

1. INTRODUCCIÓN

- XGBoost puede realizar regresión, clasificación binaria y multiclase, adicionalmente es capaz de aprender a hacer ranking,
- rápido, funciona muy bien para muestras muy grandes (billones de observaciones), además de encontrarse disponible libremente para diversas plataformas (C, C++, R, Python, Julia, entre otros)

2. GRADIENT BOOSTING

En Gradient Boosting el problema de minimización es:

$$\hat{f} = \arg\min_{f} L(f) \tag{1.1}$$

$$L(f) = \sum^{N} L\left(y_i, f\left(x_i\right)\right) \tag{1.2}$$

$$L^{(m)} \simeq \sum_{i=1}^{N}\left[l\left(y_i, \hat{y}^{(m-1)}\right) + g_i f_m\left(x_i\right) + \frac{1}{2} h_i f_m^2\left(x_i\right)\right] \tag{1.3}$$

2. GRADIENT BOOSTING

Donde $l\left(y_i, \hat{y}^{(m-1)}\right)$ es una constante, g_i el gradiente y h_i el hessiano. Sustituimos $f_m\left(x_i\right)$ por γ_{jm} y derivamos respecto γ_{jm} e igualamos a 0 para minimizar:

$$\partial L / \partial \gamma = \sum_{i=1}^{N} g_i + \sum_{i=1}^{N} h_i \gamma_{jm} = 0 \qquad (1.4)$$

Despejando γ:

$$\gamma_{jm} = \frac{\sum_{i=1}^{N} -g_i}{\sum_{i=1}^{N} h_i} \qquad (1.5)$$

2. GRADIENT BOOSTING

Para el caso de regresión la función de costo empleada es:

$$L(f) = \frac{1}{2}[y_i - f(x_i)]^2 \qquad (1.6)$$

Y su correspondiente gradiente g y hessiano h son:

$$g_i = -(y_i - f(x_i)) \qquad (1.7)$$

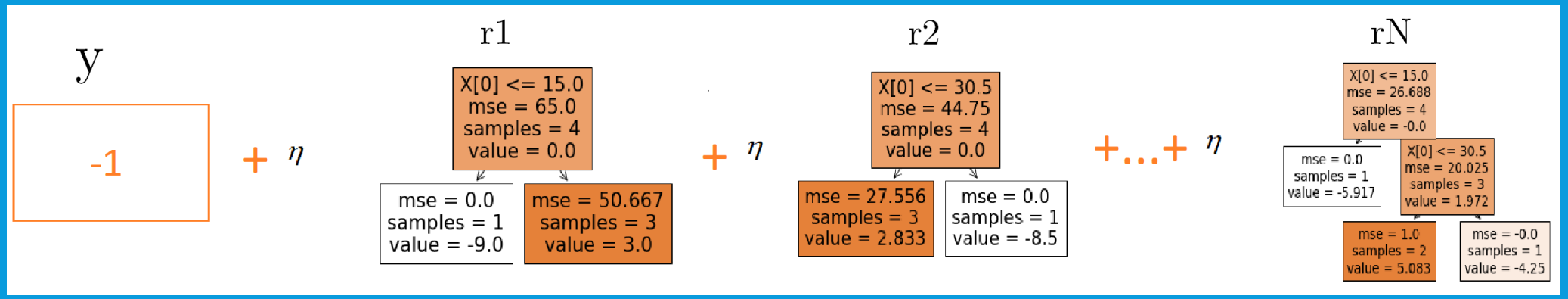
$$h_i = 1 \qquad (1.8)$$

2. GRADIENT BOOSTING

Algoritmo 1. Gradient Boosting

1. Initialize $f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$
2. For m=1 to M:
   a. For $i = 1, 2, ..., N$ compute
      $r_{im} = -[g]_{f=f_{m-1}}$
   b. Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, ..., J_m$
   c. For $j = 1, 2, ..., J_m$ compute
      $\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$
   d. Update
      $f_m(x) = f_{m-1}(x) + \eta\left(\sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})\right)$
3. Output $\hat{f}(x) = f_M(x)$

2. GRADIENT BOOSTING



2. GRADIENT BOOSTING

- Ejemplo: Ejem_Mart_GB.ipynb

3. XGBOOST

Algunas de las características que hacen diferente a XGBoost del modelo original (GradientBoost) es que introduce un término de regularización $\Omega(f_m)$ en la función de costo (1.2) convirtiéndola en:

$$L(f_m) = \sum_{i=1}^{N} L(y_i, f_m(x_i)) + \Omega(f_m) \qquad (1.12)$$

Donde:

$$\Omega(f_m) = \frac{1}{2}\lambda \sum_{j=1}^{J} \gamma_{jm} + \tau J \qquad (1.13)$$

3. XGBOOST

El cual transforma la ecuación (1.10) en:

$$\gamma_{jm} = \frac{\sum_{i \in I_j} -g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{1.14}$$

Sustituyendo (1.14) en la función de costo obtenemos el correspondiente optimo:

$$L(\gamma_{jm}) = \frac{1}{2} \sum_{j=1}^{J} \frac{\left(\sum_{i \in I_j} -g_i \right)^2}{\left(\sum_{i \in I_j} h_i \right) + \lambda} + \tau J \tag{1.15}$$

3. XGBOOST

$$L_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} -g_i\right)^2}{\left(\sum_{i\in I_L} h_i\right)+\lambda} + \frac{\left(\sum_{i\in I_R} -g_i\right)^2}{\left(\sum_{i\in I_R} h_i\right)+\lambda} - \frac{\left(\sum_{i\in I} -g_i\right)^2}{\left(\sum_{i\in I} h_i\right)+\lambda}\right] - \tau \quad (1.16)$$

3. XGBOOST

- Ejemplo: XGBOOST.ipynb

3. XGBOOST

- Otra de las diferencias con GradientBoost en la construcción de los árboles es que tiene la opción de que en lugar de buscar entre todos los puntos posibles de división lo puede hacer de manera más eficiente (sobre todo para situaciones donde la cantidad de datos sobrepasa la cantidad de memoria) proponiendo puntos para hacer las divisiones de acuerdo a percentiles de la distribución de las variables.

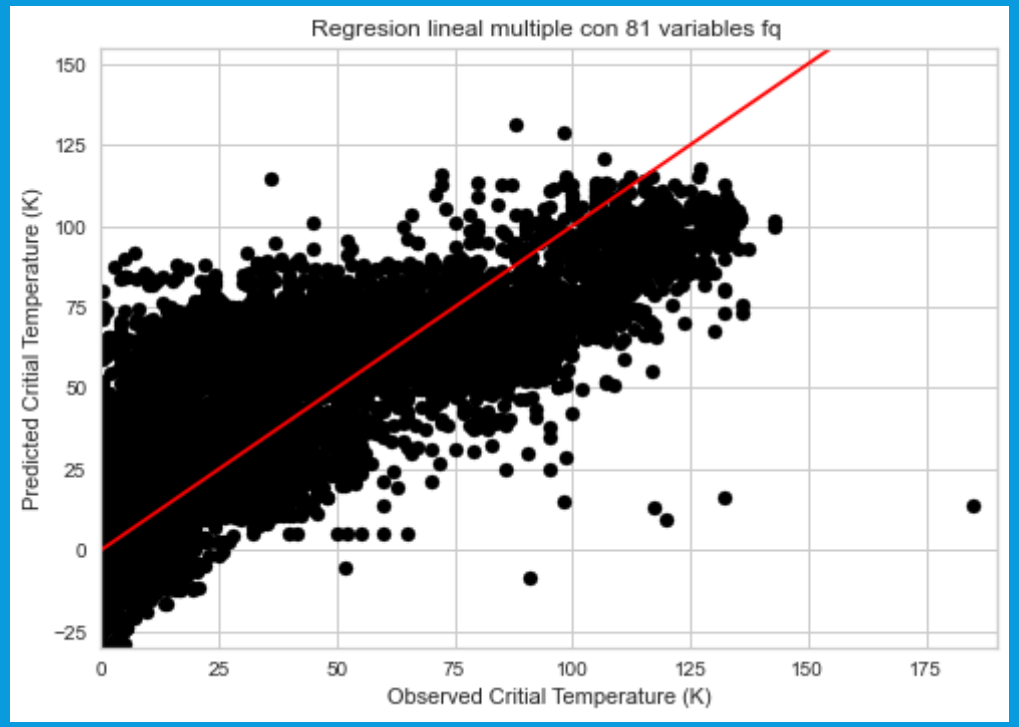
3. XGBOOST

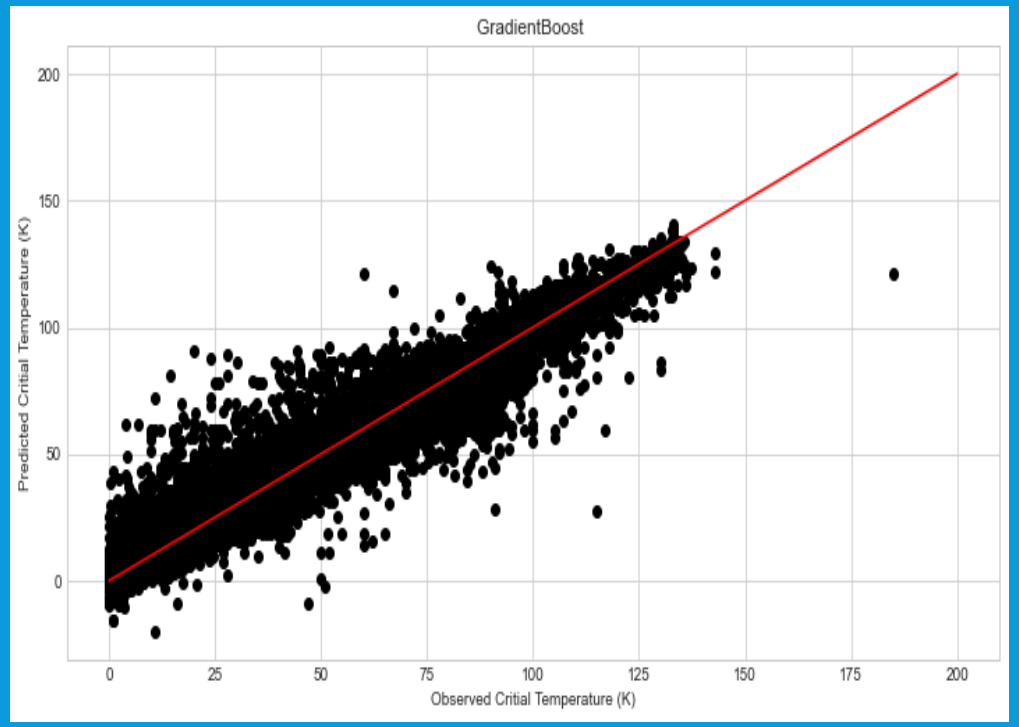
- para las observaciones que tienen datos faltantes en alguna variable o ceros, por ejemplo, propone una dirección predeterminada en los árboles que aprende de los datos
- tiene otras características que ayudan a optimizar como se utilizan los recursos de la maquina como por ejemplo guarda los datos en bloques de memoria, partes del código que son paralelizadas, además de hacer uso de la memoria cache dentro del CPU

4. APLICACIONES

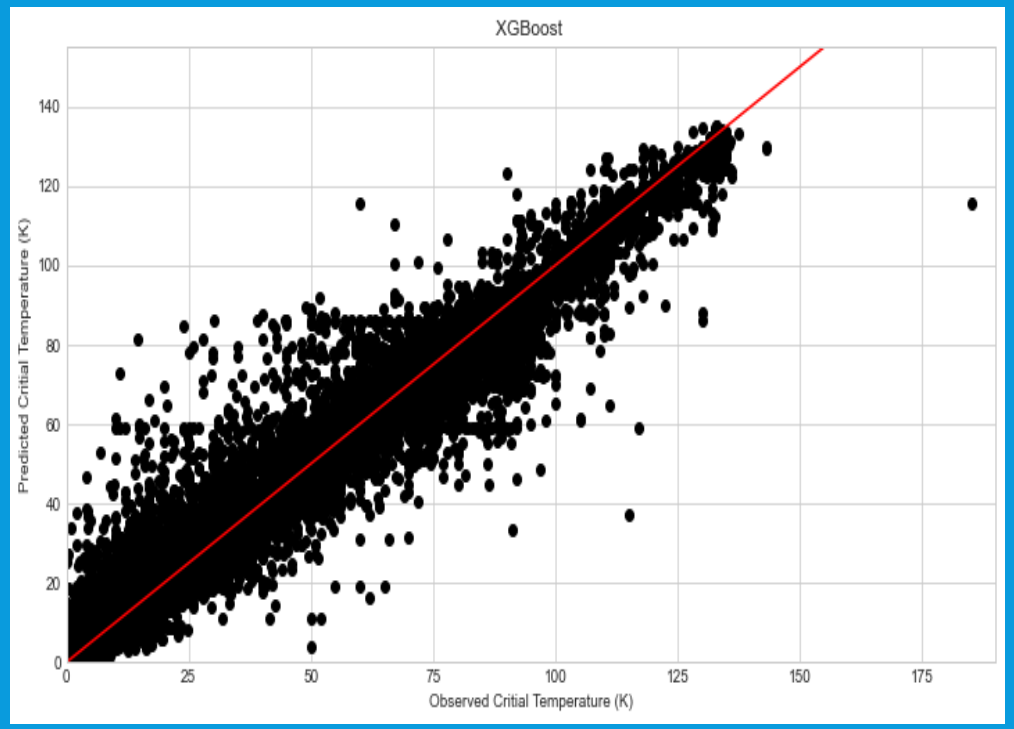
- Datos del Machine Learning Repository
- Regresión: Superconductivity Data Set
- X(21263,80), y(21263), variables numéricas
- Test set 20%
- Objetivos:
- Predecir la temperatura crítica de los materiales superconductores.
- Comparar la raíz del error cuadratico medio (RMSE), R2 y tiempo de ejecución de los modelos: regresión lineal, GradientBoost (scikit-learn), XGBoost. (k-fold=5)

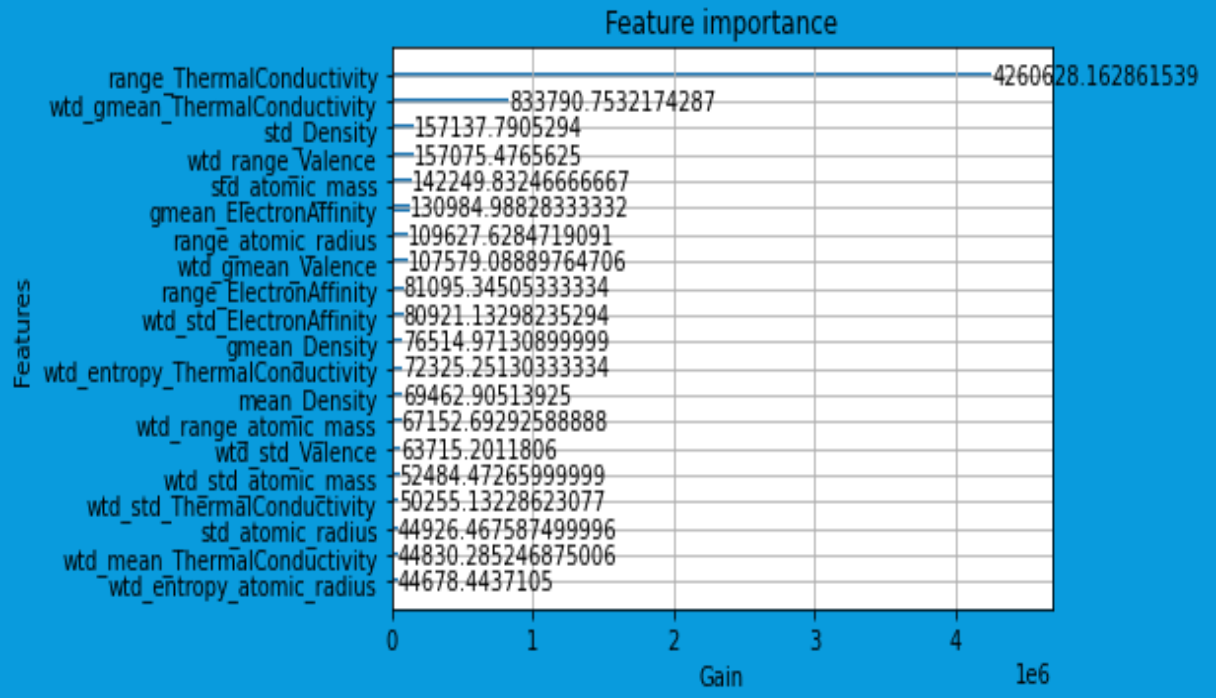
4.1 REGRESIÓN



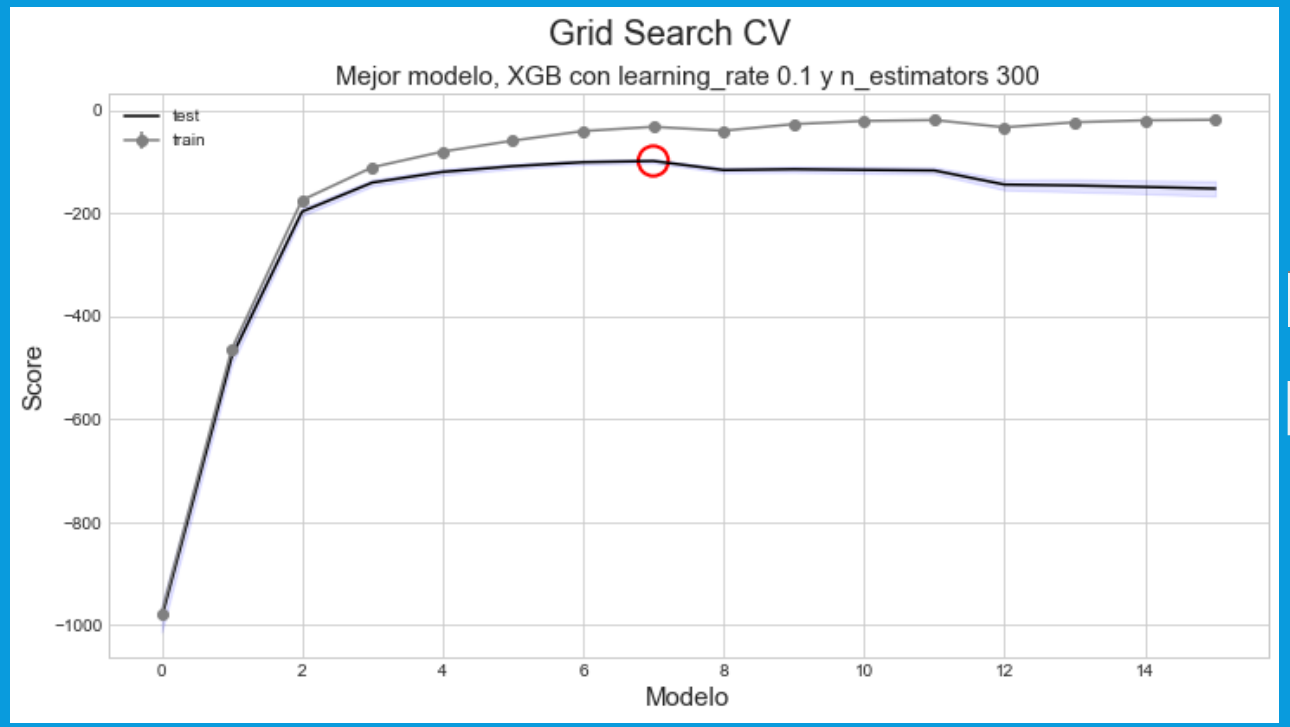


4.1 REGRESIÓN





4.1 REGRESIÓN

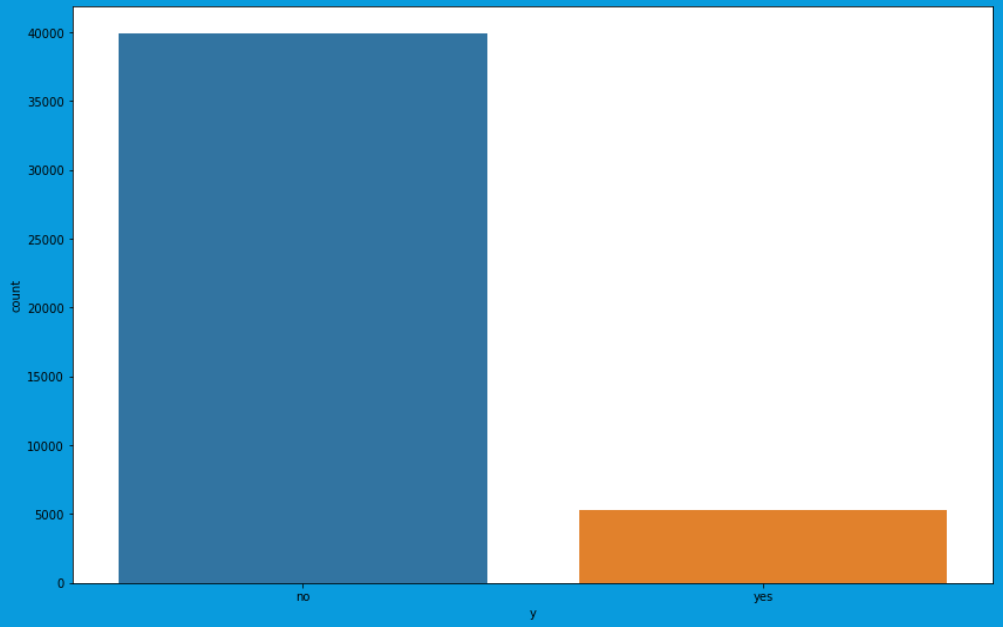


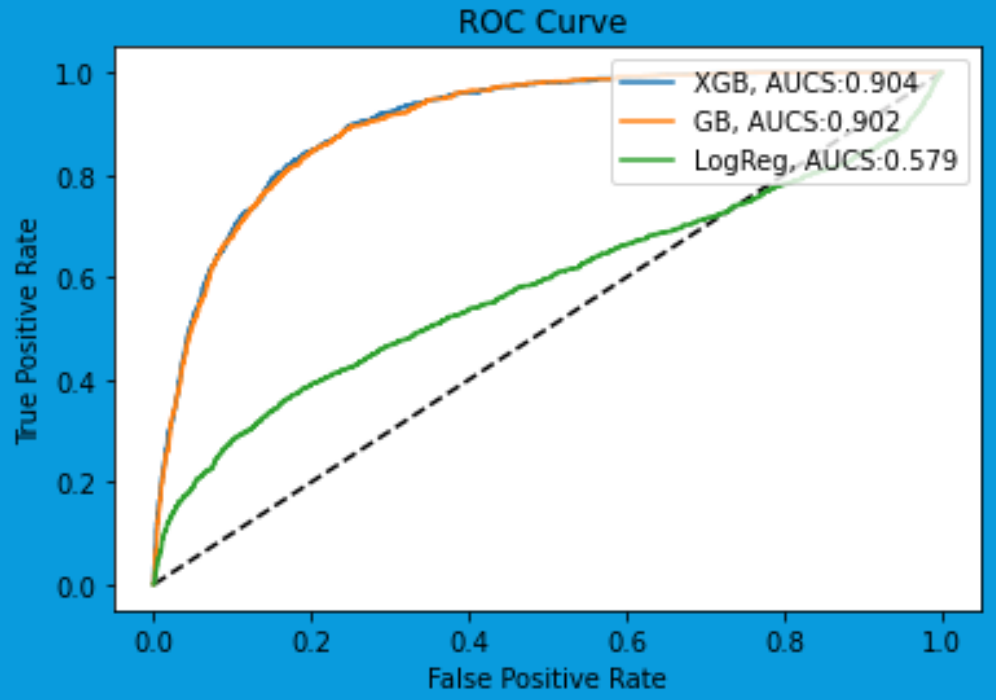
	T	RMSE	R2
LR	01.6s	17.65	0.74
GB	16min 23s	10.21	0.91
XGB	10min 15s	8.98	0.93

4. APLICACIONES

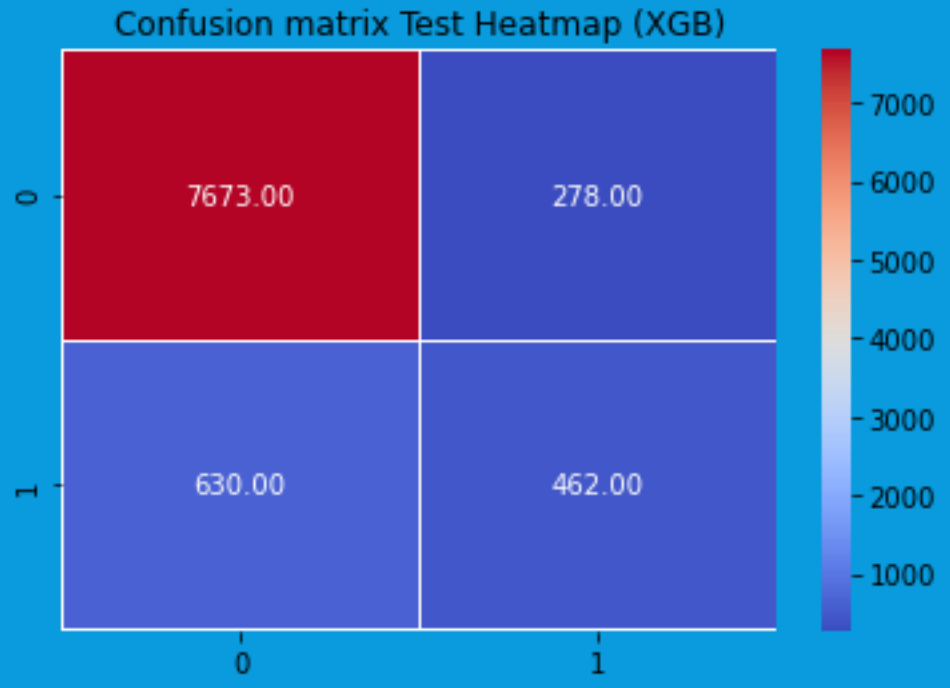
- Datos del Machine Learning Repository
- Clasificación binaria: Bank Marketing Data Set
- X(45211,16), y(45211), variables numéricas y categóricas.
- Test set 20%
- Objetivos:
- Predecir si los clientes contratarán un producto.
- Comparar el f1 score, AUC score y tiempo de ejecución de los modelos: regresión logistica, GradientBoost (scikit-learn), XGBoost. (k-fold=5)

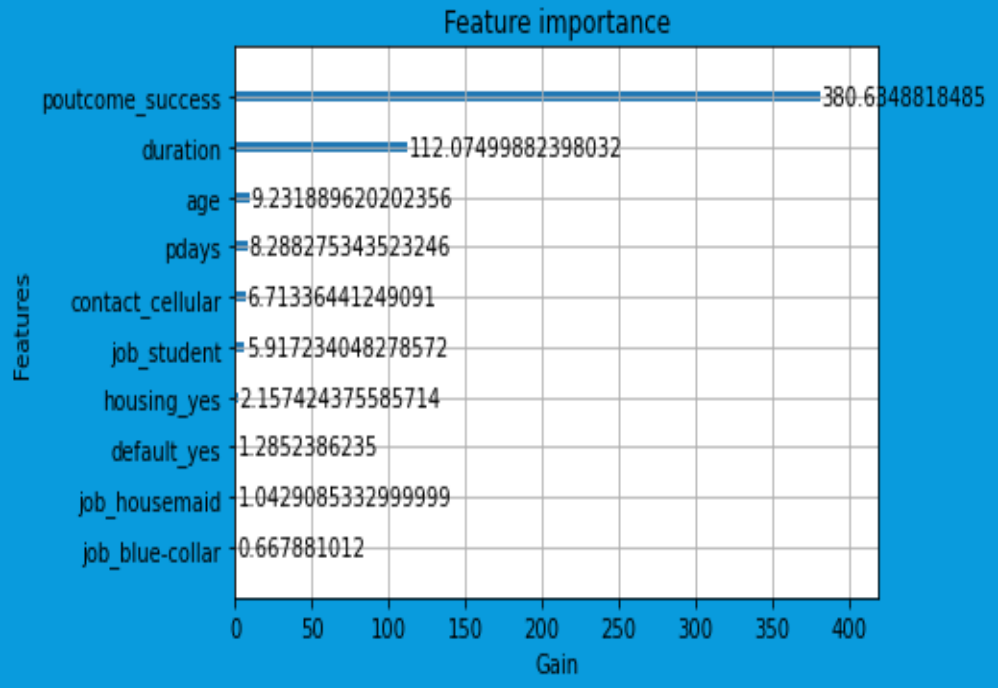
4.2 CLASIFICACIÓN



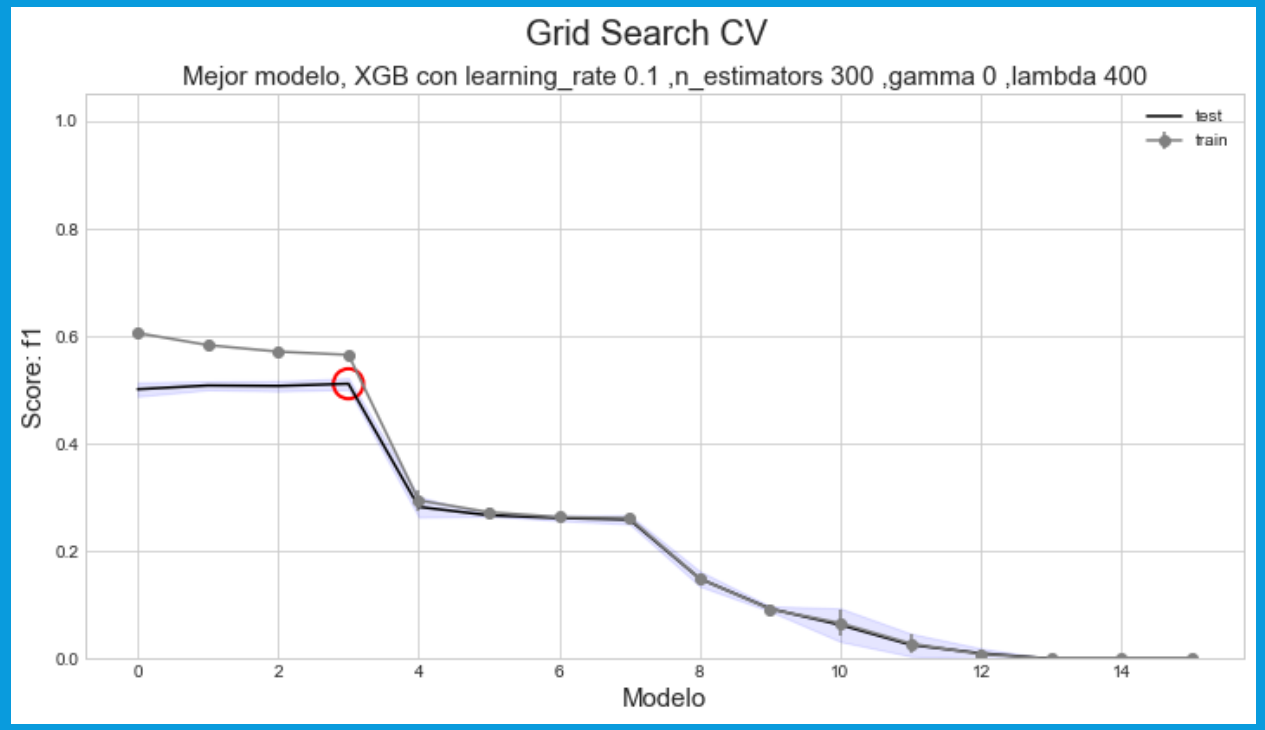


4.2 CLASIFICACIÓN





4.2 CLASIFICACIÓN



	T	F1	AUC
LOG R	3min 02.7s	0.02	0.579
GB	7min 14s	0.49	0.902
XGB	9min 16s	0.51	0.904

5. CONCLUSIONES

- XGBoost fue el mejor modelo tanto como para regresión como para clasificación, dejando lejos a los modelos de regresión lineal y logística respectivamente.
- XGBoost resultó el modelo más rápido para el caso de regresión.
- Mientras que Gradient Boost resultó el mas rápido para el caso de clasificación. (Posiblemente XGBoost fue mas lento debido al paremetro de regularización ‘gamma’).

6. REFERENCIAS

- Chen, T., and Guestrin, C. (2016), "XGBoost: A scalable tree boosting system," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-Augu, 785–794. https://doi.org/10.1145/2939672.2939785.
- Hamidieh, K. (2018), "A data-driven statistical model for predicting the critical temperature of a superconductor," *Computational Materials Science*, 154, 346–354. https://doi.org/10.1016/j.commatsci.2018.07.052.
- Hastie, T. et. all. (2009), "Springer Series in Statistics The Elements of Statistical Learning," *The Mathematical Intelligencer*, 27, 83–85. https://doi.org/10.1007/b94608.
- Izenman, A. J. (2008), *Springer Texts in Statistics Alan Julian Izenman Regression , Classification , and.*