**Regresor de Aumento de Gradiente (Gradient Boosting Regressor)** de scikit-learn, diseñado para **predecir la Evaluación de Obra** (un valor continuo, probablemente entre 0 y 1) y luego utilizar esa predicción para una tarea de **clasificación binaria** (alarma: auditar o no).

Este documento explica las partes clave del código y el proceso de modelado.

**1. Objetivo del Modelo**

El objetivo principal es:

1. **Regresión:** Predecir un valor de **"Evaluación Obra"** (una puntuación de rendimiento o calidad, por ejemplo, entre 0 y 1) para una auditoría, basándose en datos históricos y características de la obra, el personal y auditorías previas.
2. **Clasificación (Alarma):** Convertir la predicción de regresión en una decisión de **alarma** binaria (0 = OK, 1 = AUDITAR). Esto se hace comparando la predicción con un umbral, donde una evaluación predicha *por debajo* del umbral dispara la alarma.

**2. Características (Variables) Utilizadas**

El modelo utiliza una mezcla de características numéricas y categóricas:

* **Resultados de Auditorías Previas:**
  + Resultados Ultima Bodega
  + Resultados Ultima Ev OT
  + Resultados Ultima AO
  + Resultados Ultima auditoria Inventario

**Roles Clave (Target Encoding): (**SmoothedTargetEncoder)

* + Gerente de Proyecto
  + Administrador de Obra
  + Oficina Tecnica
  + Jefes de Bodega
* **Características Numéricas y Conteo:**
  + Avance Fisico (Dinamico)
  + Stock (Dinamicas).
  + Cantidad de Auditorias (conteo de auditorías previas para esa obra).
  + Cantidad Inv Generales Previos.
* **Características Categóricas (One-Hot Encoding):**
  + Empresa
  + Auditor

**3. Preprocesamiento de Datos (Pipeline) ⚙️**

El código define un ColumnTransformer dentro de un Pipeline para manejar los diferentes tipos de características:

**3.1. Imputación y Escalado de Variables Numéricas**

* **Variables Numéricas Generales (num\_cols):**
  + Se utiliza un SimpleImputer con estrategia **median** (mediana) para rellenar los valores faltantes (NaN).
  + *Nota:* No se aplica escalado (MinMaxScaler) a estas columnas, solo imputación.
* **Resultado Última Bodega (Resultados Ultima Bodega):**
  + Se aplica una tubería separada: SimpleImputer con mediana seguido de **MinMaxScaler**. Esto escala el valor para que esté en el rango [0,1] después de imputar los faltantes.

**3.2. Codificación de Variables Categóricas**

* **Roles de Personal (roles\_te\_cols): Target Encoding Suavizado**
  + Se utiliza un transformador personalizado, el SmoothedTargetEncoder.
  + Este método reemplaza la categoría (ej. el nombre del Gerente de Proyecto) con la **media suavizada de la variable objetivo (Evaluación Obra)** para esa categoría.
  + El **suavizado (smoothing=10.0)** combina la media de la categoría con la media global. Esto es crucial para **evitar el sobreajuste** a categorías con pocos datos, haciendo el *encoding* más robusto.
* **Empresa y Auditor (cat\_ohe\_cols): One-Hot Encoding (OHE)**
  + Se utiliza OneHotEncoder para convertir las categorías en columnas binarias (0 o 1).
  + Se configuran para manejar categorías nuevas (handle\_unknown="infrequent\_if\_exist") y **agrupar las categorías menos frecuentes** (aquellas con menos de 5 apariciones, min\_frequency=5) para reducir la dimensionalidad.

**4. Modelo de Regresión y Entrenamiento 🧠**

El núcleo del modelo es un **GradientBoostingRegressor** de scikit-learn.

* **Modelo:** GradientBoostingRegressor
* **Hiperparámetros:**
  + learning\_rate=0.03: El tamaño del paso de corrección, un valor pequeño hace que el aprendizaje sea más lento pero a menudo más preciso.
  + max\_depth=5: La profundidad máxima de cada árbol, limita la complejidad del modelo.
  + n\_estimators=100: El número de etapas de *boosting* (árboles a construir).
  + random\_state=42: Para reproducibilidad.
* **Estrategia de Validación Cruzada: GroupKFold**
  + Se utiliza GroupKFold con la columna de **ID de Obra (OBRA\_ID\_COL)** como grupo.
  + Esto asegura que **todas las auditorías de una misma obra caigan en el mismo pliegue (fold)** de entrenamiento o prueba, lo que evita la fuga de datos y proporciona una estimación más realista del rendimiento del modelo en obras nuevas o no vistas.

El modelo se evalúa con métricas de regresión (R², MAE, RMSE) y se entrena utilizando el *pipeline* completo.

**5. Clasificación de Alarma y Definición de Umbral 🚨**

Aunque el modelo produce un valor continuo (predicción de Evaluación Obra), la aplicación requiere una decisión binaria: **¿Auditar o no?**

1. **Etiqueta Verdadera:** La alarma real es cuando la Evaluación Obra es **menor a 0.70**. → Ytrue\_alarm​=1 si Y<0.70.
2. **Predicción de Alarma:** La alarma predicha se dispara si la predicción de regresión es menor que un umbral: → Yhat​=1 si Ypred\_reg​<Umbral.

El código evalúa la precisión (Precision), el *Recall* (Exhaustividad) y el F1-Score para una cuadrícula de umbrales (0.40 a 0.80).

* **Umbral Final:** El umbral se elige para **garantizar un *recall* de al menos 0.70** (70%). Un *recall* alto es crucial en esta aplicación, ya que significa que el modelo es bueno para **identificar una alta proporción de las obras que realmente requieren auditoría** (Minimizar Falsos Negativos).
* Si no se alcanza Recall≥0.70, se utiliza el umbral que maximiza el F1-Score.

Finalmente, el modelo se utiliza para:

* Generar el **estado del semáforo** (Verde ≥0.85, Amarillo ≥0.60, Rojo <0.60).
* Generar el **indicador binario de alarma** (alarma\_<70%) basado en el umbral final (UmbralRecall70​).

=== Clasificación de alarma (<70%) con umbral=0.710 (OOF por obra) ===

* precision recall f1-score support
* OK 0.61 0.42 0.49 195
* AUDITAR 0.54 0.72 0.62 186
* accuracy 0.56 381
* macro avg 0.57 0.57 0.56 381
* weighted avg 0.58 0.56 0.55 381

En resumen: El modelo está **sesgado para ser exhaustivo** en la detección de problemas (alto *Recall* para "AUDITAR"), lo cual es a menudo deseable en tareas de alerta, a expensas de un mayor número de falsas alarmas (baja *Precision* para "AUDITAR").