**Pasos para resolver Credit Card Fraud Detection**

**El problema**

El dataset contiene **transacciones con tarjeta de crédito** realizadas en dos días en Europa.  
El reto es **detectar cuáles transacciones son fraudulentas** en un escenario altamente **desbalanceado**:

* Transacciones normales: ~284,000 (99.83%).
* Transacciones fraudulentas: ~492 (0.17%).

Esto refleja la realidad: los fraudes son raros pero críticos de detectar.

**Variables del dataset**

* **Time** → segundos transcurridos desde la primera transacción en el dataset.
* **Amount** → valor monetario de la transacción.
* **V1 a V28** → variables numéricas obtenidas con un **Análisis de Componentes Principales (PCA)** sobre las características originales (para proteger la confidencialidad de los clientes y comerciantes).
* **Class** → variable objetivo (etiqueta):
  + 0 = transacción legítima.
  + 1 = transacción fraudulenta.

**Qué queremos predecir**

El objetivo es **construir un modelo que prediga si una transacción es fraude (Class = 1) o no fraude (Class = 0)**.

En otras palabras: dado el **monto, el tiempo y las 28 variables transformadas**, queremos asignar una probabilidad de fraude a cada transacción.

**1. Exploración inicial**

* Descargar el dataset de Kaggle (link).
* Revisar estructura:
  + 284,807 transacciones.
  + Variables transformadas con PCA (V1…V28).
  + Amount (monto) y Time.
  + Class (0 = no fraude, 1 = fraude).

Problema: **altamente desbalanceado** (~0.17% de fraudes).

*import kagglehub*

*# Download latest version*

*path = kagglehub.dataset\_download("mlg-ulb/creditcardfraud")*

*print("Path to dataset files:", path)*

**Enfoque A: Usar GANs para generar datos sintéticos (Oversampling)**

Objetivo: aumentar los casos de fraude en el dataset, porque son muy pocos.

**Pasos:**

1. **Analizar el dataset**
   * Confirmar el desbalance: hay ~99.8% transacciones normales y ~0.2% fraudes.
2. **Preprocesar los datos**
   * Escalar variables (Amount, Time).
   * Mantener las variables transformadas (V1…V28).
3. **Entrenar una GAN tabular (ej. CTGAN o CopulaGAN)**
   * La GAN aprende la distribución de los datos, en particular de la clase minoritaria (fraudes).
   * Se entrena de manera condicional para generar ejemplos con Class = 1 (fraude).
4. **Generar muestras sintéticas de fraudes**
   * Crear suficientes ejemplos fraudulentos para reducir el desbalance (ej. pasar de 0.2% → 5% o 10%).
5. **Combinar dataset real + sintético**
   * Mantener los datos de test **sin tocar**, solo usar los sintéticos en entrenamiento.
6. **Entrenar un clasificador tradicional**
   * Usar modelos como XGBoost, LightGBM o RandomForest.
   * Ahora el dataset está más balanceado y el modelo aprende mejor.
7. **Evaluar desempeño**
   * Usar métricas adecuadas: Recall, Precision, F1-Score y PR-AUC.
   * Validar que el modelo no dependa demasiado de los datos sintéticos (verificar distribuciones).

**Enfoque B: Usar GANs como detectores de anomalías**

Objetivo: detectar fraudes porque **no se parecen a las transacciones normales**.

**Pasos:**

1. **Separar datos normales (Clase = 0)**
   * Entrenar la GAN solo con transacciones normales.
2. **Entrenar la GAN como reconstructor**
   * El **generador** intenta recrear las transacciones normales.
   * El **discriminador** evalúa qué tan realistas son.
3. **Medir error de reconstrucción**
   * Como la GAN aprendió solo de normales, reconstruirá bien datos normales.
   * Pero **los fraudes se reconstruyen mal**, porque son “raros” para la GAN.
4. **Definir un “anomaly score”**
   * Combinar el error de reconstrucción con la confianza del discriminador.
   * Cuanto más alto sea el score, más probable es que sea fraude.
5. **Ajustar umbral**
   * Usar un conjunto de validación para elegir un umbral que logre buen Recall (detectar la mayoría de fraudes) y un Precision aceptable.
6. **Evaluar en test**
   * Medir PR-AUC y Recall en fraudes.
   * Ajustar parámetros hasta balancear detección vs falsos positivos.

**Enfoque C: Híbrido (lo más fuerte en práctica)**

* **Parte 1:** Usar **CTGAN** para oversampling y entrenar un buen clasificador supervisado.
* **Parte 2:** Entrenar una **GAN de anomalías** sobre transacciones normales.
* **Parte 3:** Combinar ambos:
  + Si el clasificador predice fraude **o** el detector de anomalías da score alto → marcar como sospechoso.

Esto mejora la detección de fraudes **conocidos** (con etiquetas) y también de **nuevos patrones** de fraude.

**Métricas clave (importante en GANs para fraude)**

* **Precision-Recall AUC** → mejor que ROC-AUC en datasets desbalanceados.
* **Recall (Sensibilidad):** % de fraudes detectados.
* **Precision:** % de alertas que realmente eran fraude.
* **Costo esperado:** (#fraudes no detectados × costo fraude) + (#falsos positivos × costo de investigar).

**Resumen en pasos simples**

1. Entender el dataset y su desbalance.
2. Escoger enfoque: **Oversampling con GAN tabular** o **Anomaly detection con GAN**.
3. Entrenar GAN:
   * Si es oversampling → generar fraudes sintéticos.
   * Si es anomalía → entrenar con normales y calcular score.
4. Integrar resultados: entrenar clasificador o usar anomaly score.
5. Evaluar con PR-AUC, Recall y costos.
6. Ajustar umbrales y validar.
7. Desplegar y monitorear.