CREDIT FRAUD DETECTION

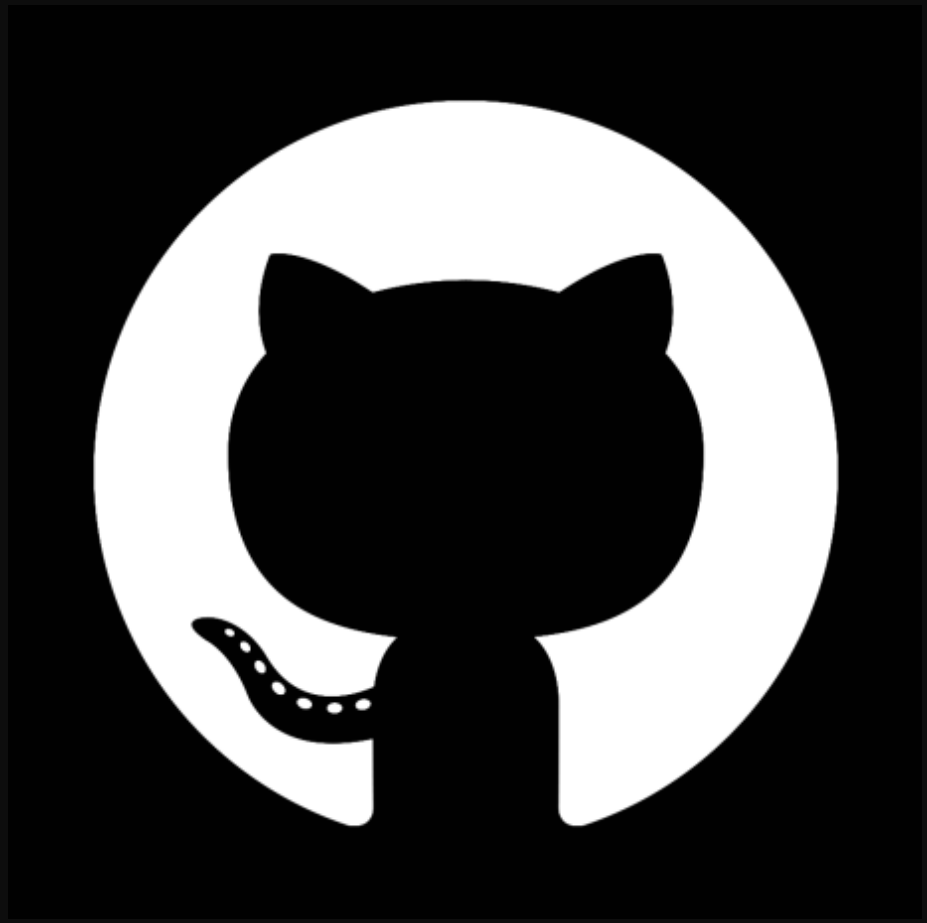
Juan Jacobo Puente

Rodion Bondarets

Victor Casas

Indice

Enlace al Proyecto



¿Qué es el fraude con tarjetas de crédito?

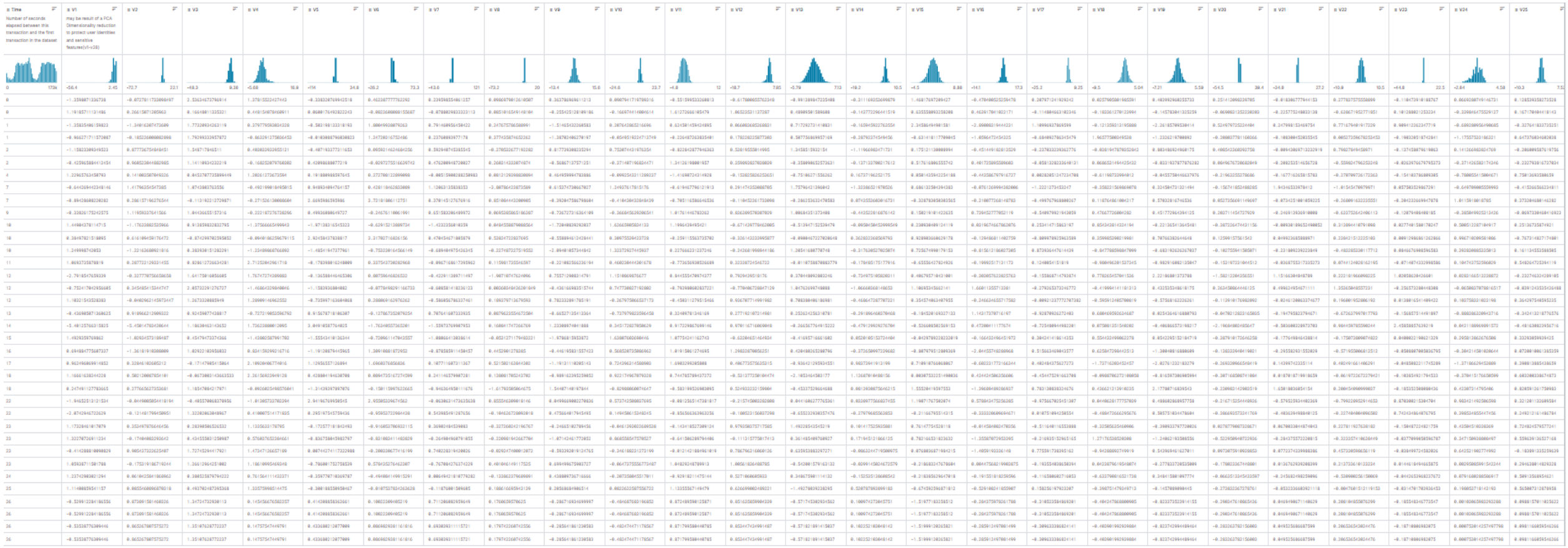
- El fraude con las tarjetas de crédito ocurre cuando algún tercero realiza una transacción sin ninguna autorización.

- Nuestros modelos tendrán como objetivos identificar nuevas transacciones para minimizar estos costes a la empresa.
- Queremos detectar el mayor numero de transacciones fraudulentas y minimizar los falsos positivos.

Recoleccion de datos

- Los datos han sido recolectados de Kaggle.
- (https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)
- Tiene 31 columnas con 284807 registros cada una.
- “Class” es la variable a estudiar.

Dataset



EDA

- Las variables de V1 a V28 han sido el resultado de un PCA por términos de privacidad.
- Amount: Cantidad de transacciones.
- Time: Segundos que han pasado entre transacción.
- Class: variable objetivo. Estado de la transacción.

EDA

- Registros de transacciones no fraudulentas(Class): 284315
- Registros de transacciones frauduletentas(Class): 492
- No hay registros nulos ni duplicados.

Preparación de los datos

- Escalamos las variables “Time” y “Amount”.
- Eliminación de outliers manteniendo todos los de clase minoritaria de fraude(“Class=1”).

Preparación para el Modelado

División de Datos	Balanceo de Clases	Selección de Características
Utilizamos **train_test_split** para dividir los datos en conjuntos de **entrenamiento, prueba y validación**.	Calculamos **scale_pos_weight** para manejar el desbalance de clases en los datos.	Todas las variables, excepto '**Class**', se utilizan como características para el modelo.

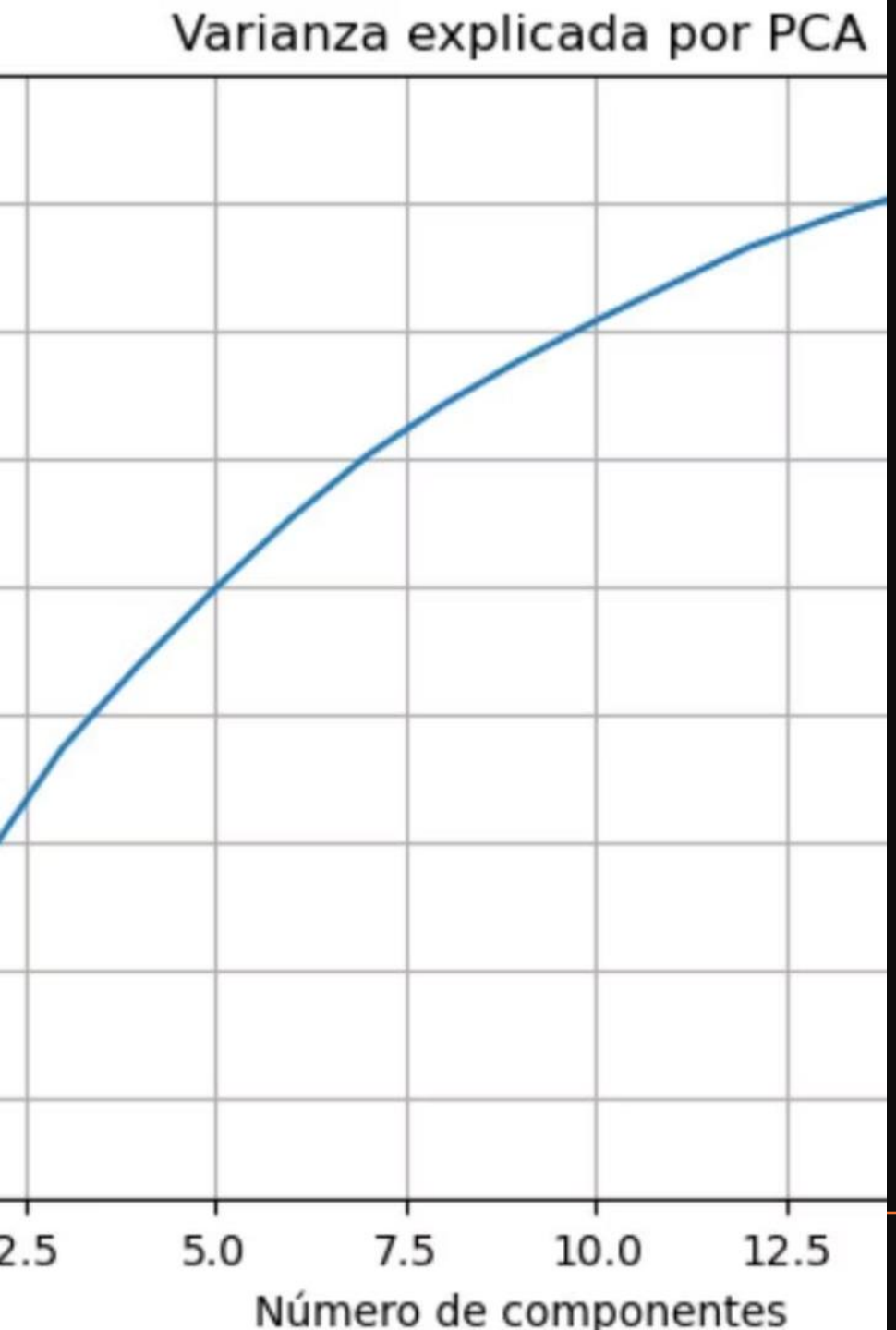
Selección de modelos

- **XGBoost** destaca por su robustez ante datos desbalanceados y capacidad para manejar variables complejas, siendo especialmente potente en la captura de patrones no lineales en transacciones fraudulentas.

- **LightGBM** ofrece un rendimiento similar con mayor eficiencia computacional, lo que nos permite procesar grandes volúmenes de transacciones con menor tiempo de entrenamiento mientras mantiene alta precisión.

- **SVC** (Support Vector Classifier) proporciona una excelente capacidad para establecer límites de decisión claros entre transacciones legítimas y fraudulentas, siendo particularmente efectivo cuando las características fraudulentas presentan separaciones complejas.

- **Regresión Logística** aporta interpretabilidad y transparencia al proceso, permitiéndonos identificar la importancia relativa de cada variable en la detección de fraude, aspecto crucial para explicar resultados a stakeholders no técnicos."



Preprocesamiento de datos y PCA

El conjunto de datos original se limpió eliminando los casos atípicos, salvo en los casos de fraude. Las funciones de tiempo y cantidad se escalaron por separado. PCA se aplicó para reducir la dimensionalidad mientras se preserva una varianza del 95%, lo que dio como resultado 19 componentes principales utilizados para el modelado.

Los datos se dividieron en conjuntos de capacitación, validación y pruebas con estratificación para mantener las proporciones de la clase, asegurando una evaluación equilibrada.

Modelo XGBoost Inicial

Configuración del Modelo	Entrenamiento	Evaluación
Implementamos un clasificador XGBoost con parámetros iniciales, incluyendo scale_pos_weight para manejar el desbalance de clases.	El modelo se entrena con los datos de entrenamiento utilizando la métrica de evaluación AUC.	Se realizan predicciones en el conjunto de prueba y se evalúan utilizando matriz de confusión, informe de clasificación y puntuación ROC AUC.

Optimización de Hiperparámetros

1

Búsqueda Aleatoria

Realizamos una búsqueda aleatoria para explorar diferentes combinaciones de hiperparámetros.

2

Búsqueda en Cuadrícula

Refinamos la búsqueda utilizando GridSearchCV para encontrar la mejor combinación de parámetros.

3

Evaluación del Mejor Modelo

Evaluamos el rendimiento del modelo optimizado utilizando métricas de clasificación y ROC AUC.

Aplicacion SMOTE

Oversampling

Aplicamos SMOTE (Synthetic Minority Over-sampling Technique) para equilibrar las clases en el conjunto de entrenamiento.

Análisis

Determinamos que el modelo sin SMOTE muestra mejores resultados generales.

Nuevo Modelo XGBoost

Entrenamos un nuevo modelo XGBoost con los datos balanceados.

Comparación de Resultados

Evaluamos y comparamos el rendimiento del modelo con y sin SMOTE.

Optimización de Hiperparámetros

1

Búsqueda Aleatoria
Realizamos una búsqueda aleatoria para explorar diferentes combinaciones de hiperparámetros.

2

Búsqueda GridSearch
Refinamos la búsqueda utilizando GridSearchCV para encontrar la mejor combinación de parámetros.

3

Evaluación del Mejor Modelo
Evaluamos el rendimiento del modelo optimizado utilizando métricas de clasificación y ROC AUC.

Support Vector Classifier con muestreo aleatorio bajo

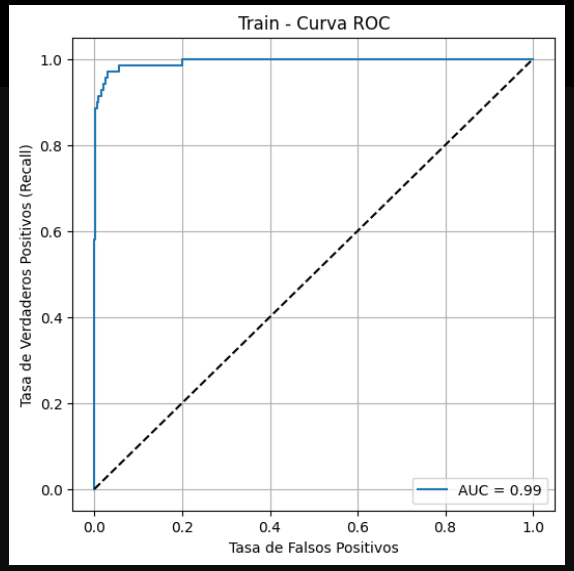
Preparación de datos

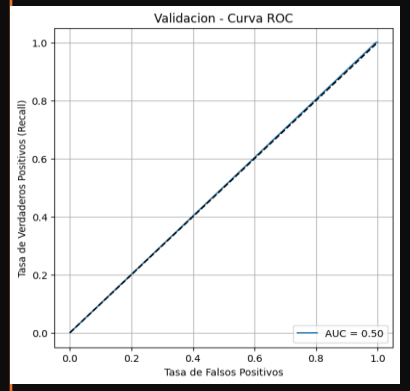
Se utiliza el muestreo aleatorio para equilibrar las clases en los datos de formación. Las características se han escalado con RobustScaler. Estratificado K-Fold cross-validación y GridSearchCV hiperparámetros optimizados.

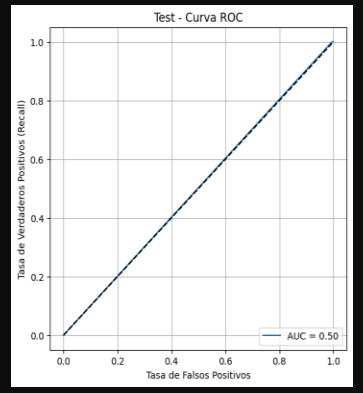
Rendimiento del modelo

Mejores parámetros: C=1, kernel=lineal. El AUC de entrenamiento fue de 0,95 con una precisión del 96%. Las puntuaciones de la AUC de prueba y validación fueron cercanas a 0,5, lo que indica una mala generalización. Las puntuaciones F1 para la clase minoritaria eran muy bajas.

Support Vector Classifier sin muestreo







Enfoque

SVC entrenado con class_weight='equilibrado' y núcleo polinomial sobre datos desequilibrados originales sin muestreo.

Resultados

El AUC de entrenamiento fue de 0,99, clara indicacion de overfitting, con una precisión del 99,9%. Sin embargo, el AUC de prueba y validación cayó a 0,5 con una precisión cercana a cero, lo que indica que el modelo no pudo generalizar y predijo principalmente la clase mayoritaria.

Clasificador LightGBM sin muestreo

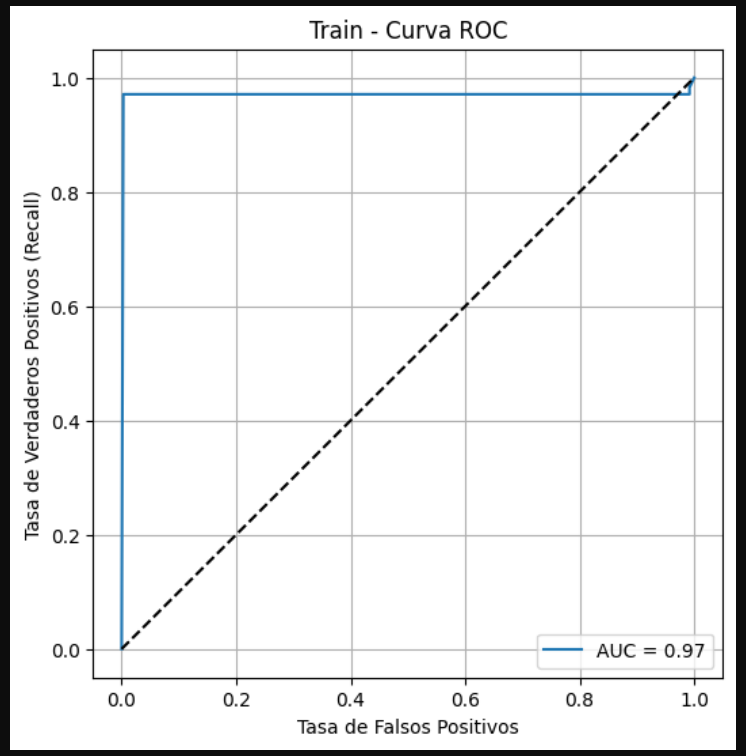
Configuración del modelo

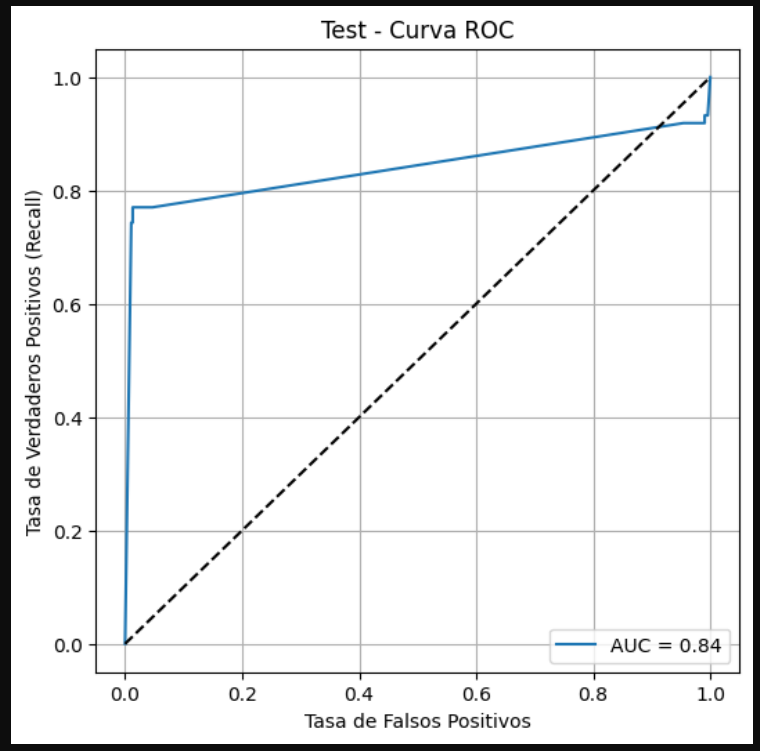
Utiliza LightGBM con escale_pos_weight para manejar el desequilibrio de clase. La búsqueda de cuadrícula optimiza parámetros como la tasa de aprendizaje, profundidad máxima y número de hojas. RobustScaler se aplicó a las características.

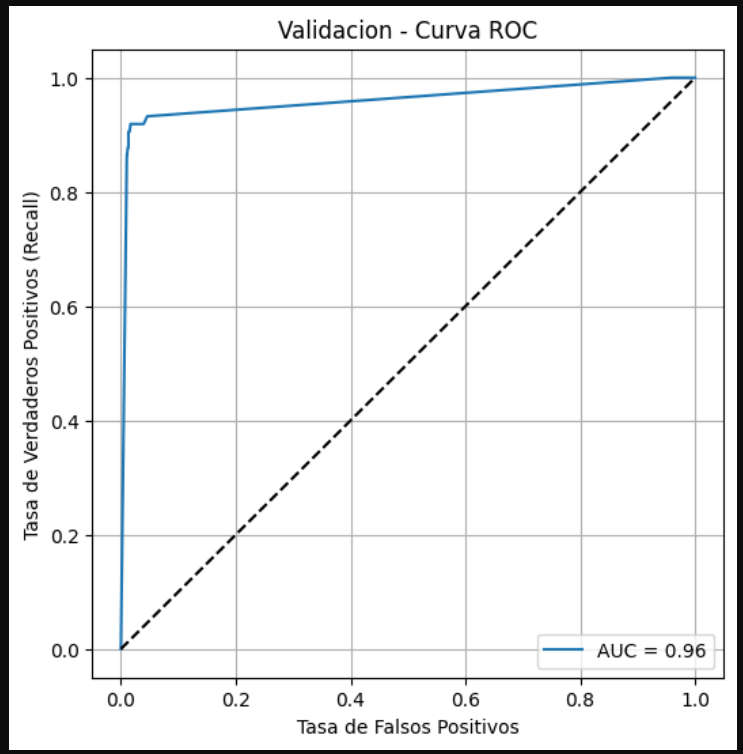
Rendimiento

El AUC de entrenamiento fue de 0,98 con una precisión del 99,7%. Los AUCs de prueba y validación fueron 0,88 y 0,95, respectivamente, con un alto grado de recuerdo pero baja precisión para la clase minoritaria. Las puntuaciones de F1 fueron moderadas, mostrando una mejor generalización que la VCS.

Clasificador LightGBM sin muestreo







Comparación de modelos clásicos: regresión logística, árbol de decisión, bosque aleatorio

Regresión logística

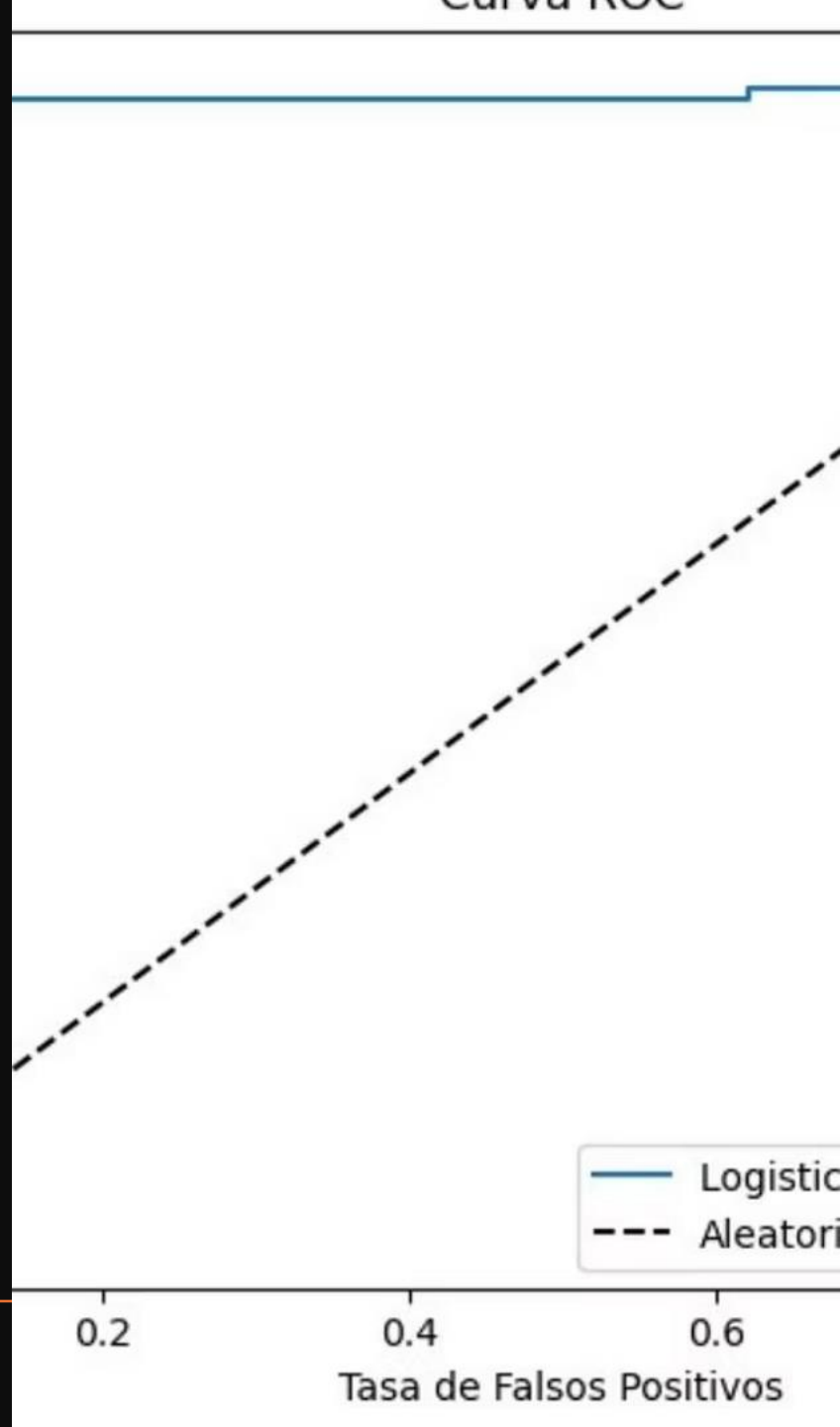
Alta precisión (~99%) y AUC (0,99), pero baja precisión (0,22) y moderada F1 (0,35) para la clase de fraude.

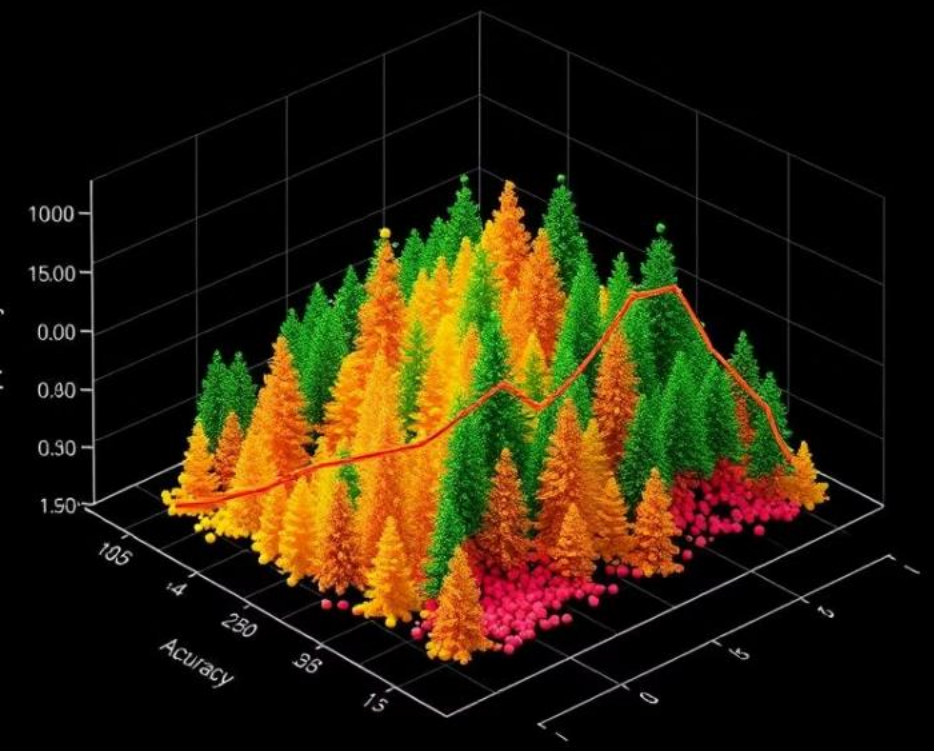
Árbol de decisiones

Mejora del recuerdo (0,93) y F1 (0,69) para el fraude, con una precisión de 99,7% y un AUC de 0,94.

Random Forest

Mejor en general con un 99,9% de precisión, 0,98 de recuperación y 0,98 F1 para fraude. AUC casi perfecta (0,995).

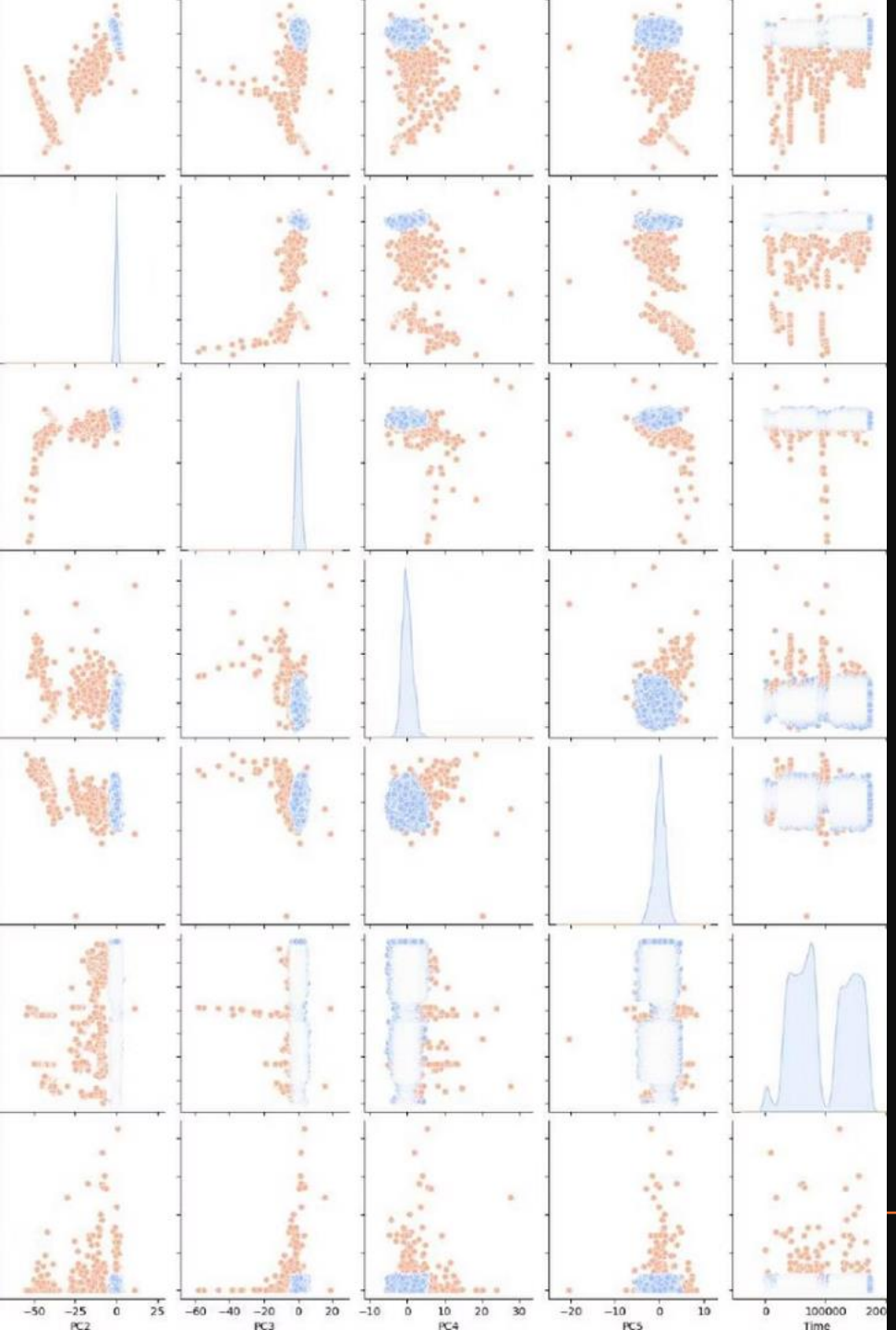




Random Forest Hyperparameter Tuning

La búsqueda de cuadrículas probó varios valores n_estimators, max_depth y min_samples_split con subsampling equilibrado. Los mejores parámetros incluyen max_depth=6 o 8 y n_estimators entre 80-300.

La validación cruzada mostró puntuaciones de recuperación consistentes alrededor de 0,85-0,96, lo que indica un rendimiento robusto de detección de fraude a través de los pliegues.



LightGBM con PCA e ingeniería de características

LightGBM entrenado en datos PCA-reducidos con escale_pos_weight y parada temprana logró excelentes resultados: 99% de precisión, 0.95 recuerdo, y 0.97 F1-puntuación para la clase de fraude.

El AUC-ROC fue de 0,989 y el AUC-PR de 0,965, lo que indica una fuerte separación de clases y una detección fiable a pesar del desequilibrio.

Clasificador XGBoost con y sin sobremuestreo

Con sobremuestreo SMOTE

El modelo detectó más fraudes (alto recuerdo) pero produjo muchos falsos positivos, lo que redujo la precisión y la puntuación F1.

Sin sobremuestreo

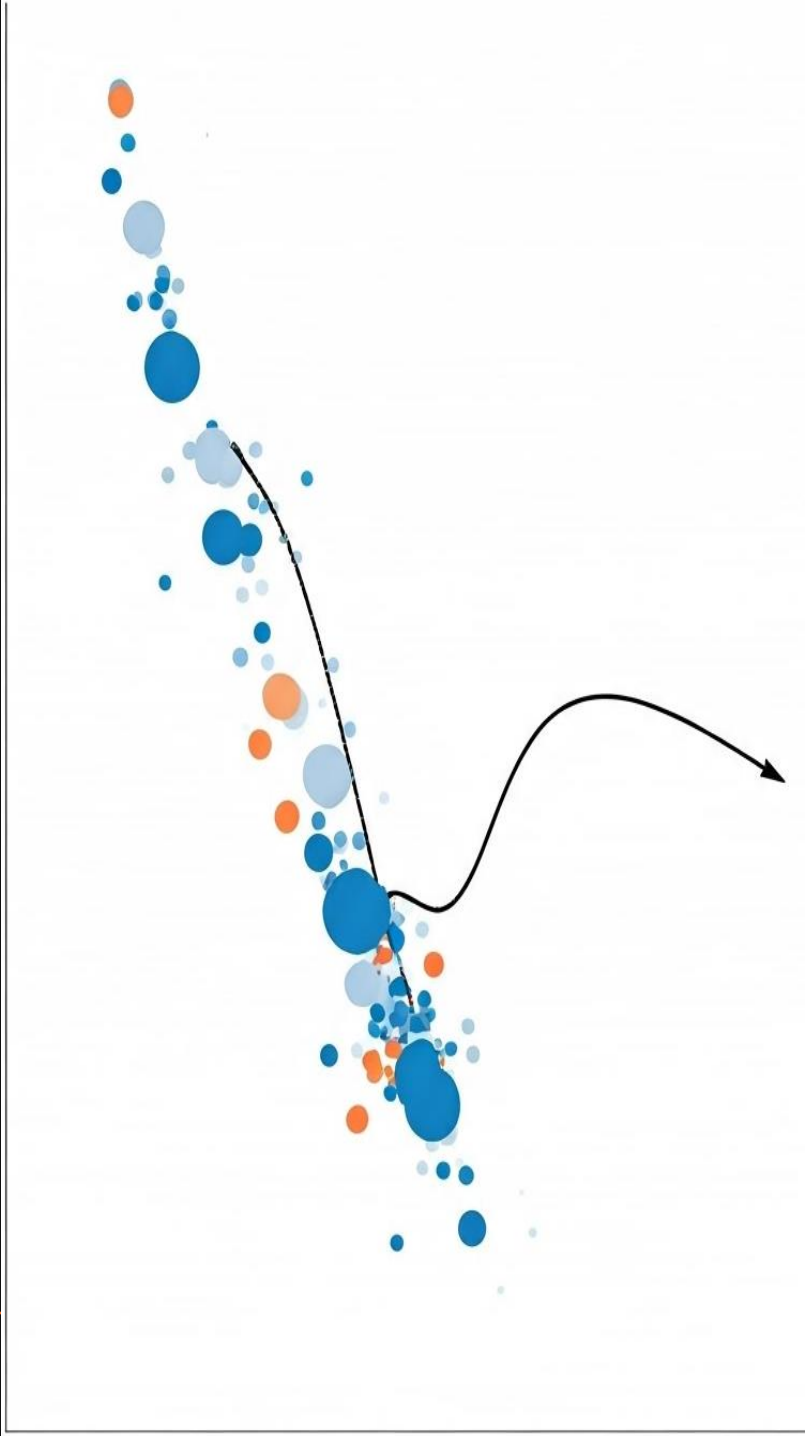
Mejor balance con 92% de recuerdo y precisión, alta precisión (99%) y AUC-ROC (0,989). La matriz de confusión mostró menos falsos positivos.

Modelo de Regresion Logistica

Implementación
Entrenamos un modelo de regresión logística como alternativa al XGBoost.

Comparación
Evaluamos el rendimiento con y sin oversampling (SMOTE).

Optimización
Realizamos una búsqueda aleatoria para optimizar los hiperparámetros del modelo.



RESULTADOS

Modelo	Dataset	AUC	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	Falsos Positivos	Falsos Negativos
XGBoost	Train	0.969	0.96	0.53	1.00	0.70	16,219	0
	Validation	0.962	1.00	0.48	0.85	0.61	69	11
	Test	0.969	1.00	0.53	0.82	0.65	54	13
LightGBM	Train	0.984	0.997	0.38	0.97	0.55	108	2
	Test	0.878	0.986	0.09	0.77	0.16	580	17
	Validation	0.946	0.986	0.10	0.91	0.18	593	7
LogReg	Test	0.957	1.00	0.80	0.59	0.68	11	30
SVC	Train	0.927	0.999	0.74	0.86	0.79	21	10
	Test	0.500	0.003	0.00	1.00	0.00	42,615	0
SVC-RUS	Train	0.951	0.962	0.04	0.94	0.08	7,566	21
	Test	0.506	0.998	0.03	0.01	0.02	29	73

Conclusiones

Ganador: XGBoost

- 1. Mejor Recall (0.82): Detecta más fraudes reales que Logistic Regression.
- 2. F1-Score decente (0.65): Mejor equilibrio que LightGBM.
- 3. AUC alto (0.9688): Buen poder discriminativo.

Mencion Honorifica: Logistic Regression

- Ventaja: Menos falsos positivos (Precision = 0.79).
- - Útil si: El negocio prefiere **evitar molestar a clientes* (menos bloqueos erróneos).

Conclusiones

Resultados Clave del LGBMClassifier que lo descalifican:

- **Precision (Clase 1 - Fraude): 0.09**
  → **Solo el 9%** de las predicciones positivas son realmente fraudes → *Alta cantidad de falsos positivos*.
- **Recall (Clase 1 - Fraude): 0.77**
  → **El modelo logra capturar 77%** de los fraudes → *Baja cantidad de falsos negativos*.
- **F1-Score (Clase 1 - Fraude): 0.16**
  → *Bajo balance* entre precisión y recall → No optimiza bien ambos a la vez.
- **AUC (Area Under Curve): 0.8783**
  → *Bastante bueno*, el modelo discrimina relativamente bien entre fraude y no fraude, aunque podría mejorar.

Conclusiones

SVC (Support Vector Classifier)

- **Métricas principales:**
  - AUC ≈ 0.5 (equivalente a adivinar al azar)
  - F1-Score (Clase 1): 0.0 (no detecta fraudes)
- **Problemas detectados:**
  - No maneja adecuadamente el desbalanceo de clases, incluso usando ajuste de pesos.

SVM con UnderSampling (RUS)

- **Métricas principales:**
  - Recall (Clase 1): 0.0 (no identifica fraudes)
  - F1-Score: 0.0
- **Problemas detectados:**
  - El undersampling eliminó demasiada información relevante, afectando el rendimiento general.

Propuestas de Mejora

Enfocarse en XGBoost o Logistic Regression:

- Ajustar hiperparámetros para mejorar Precision (XGBoost) o Recall (Logistic Regression).

Técnicas para Desbalanceo:

- *SMOTE* o *ADASYN*: Generar muestras sintéticas de la clase minoritaria.
- *Class Weight*: Aumentar el peso de la clase 1 en XGBoost (scale_pos_weight).

Ensamble:

- Combinar Logistic Regression + XGBoost en un *VotingClassifier*.

Threshold Optimization:

- Ajustar el umbral de decisión para mejorar Precision o Recall (ej.: usar curva Precision-Recall).