Detección de Fraude con Tarjetas de Crédito mediante Modelos Supervisados y Reducción de Dimensión

Yendri Ferreira, Leonardo Araque Quintero

***Resumen*—El fraude con tarjetas de crédito representa pérdidas multimillonarias para el sector financiero y deteriora la confianza de los usuarios. Este trabajo aborda el *Credit Card Fraud Detection* dataset de Kaggle (284 807 transacciones, 0.172 % fraudes) mediante un flujo de aprendizaje supervisado robusto al desbalance de clases. Se comparan cinco familias de modelos, se aplican técnicas de balanceo (SMOTE, *class-weight*) y se estudian estrategias de reducción de dimensión (RFE, PCA). El mejor modelo, un *Random Forest* balanceado, alcanza AUC 0.992 y Recall 96 % en prueba, superando técnicas no supervisadas y acercándose a arquitecturas Transformer recientes.**

ÍNDICE

***Impact Statement*—Detectar transacciones fraudulentas en tiempo real reduce pérdidas económicas y evita bloqueos injustificados que afectan la experiencia del cliente. La solución propuesta es ligera, reproducible y entrenable en hardware convencional, por lo que puede integrarse en pasarelas de pago de pequeñas y medianas instituciones latinoamericanas que carecen de infraestructura costosa.**

***Index Terms*—Detección de fraude, tarjetas de crédito, aprendizaje supervisado, desbalance de clases, SMOTE, PCA.**

I. Introducción

El crecimiento del comercio electrónico ha multiplicado tanto el volumen de transacciones con tarjeta como las oportunidades de fraude. Los sistemas basados en reglas estáticas han quedado rezagados frente a esquemas delictivos dinámicos y cada vez más sofisticados. Este trabajo propone un flujo de *machine learning* que compara y combina varios modelos supervisados, integra técnicas de balanceo y estudia la reducción de dimensión, con el fin de maximizar la sensibilidad al fraude sin sacrificar eficiencia computacional.

II. Descripción del problema

II-A. Contexto del problema

El fraude con tarjetas de crédito es un problema global para bancos, comercios y usuarios. A medida que los métodos de pago se digitalizan, los estafadores refinan sus estrategias, haciendo ineficaces los sistemas basados en umbrales fijos. Desde la perspectiva de ciencia de datos, el fenómeno es raro, etiquetado y altamente desbalanceado, lo que motiva el uso de aprendizaje automático supervisado y técnicas de muestreo.

II-B. Composición de la base de datos

Se utiliza el dataset público *Credit Card Fraud Detection* de Kaggle: 284 807 transacciones europeas en dos días, 31 columnas numéricas (`Time`, `Amount`, `Class` y `V1–V28` derivadas de PCA). No hay valores faltantes y solo el 0.172 % son fraudes (492 casos).

II-B1. Características clave:

- **Fuerte desbalance**: requiere SMOTE, undersampling o pesos de clase.
- **Ausencia de NaNs**: simplifica el preprocesamiento.
- **Baja multicolinealidad**: las variables PCA están casi ortogonales.

II-C. Análisis exploratorio resumido

Las transacciones fraudulentas muestran montos más altos (mediana €118) y dispersos que las legítimas (mediana €22). El histograma de `Amount` presenta una cola larga; `Time` es casi uniforme.

La Fig. 1 muestra que las componentes `V10`, `V14` y `V17` presentan la correlación más alta con la clase fraudulenta, justificando su consideración en la selección de características.

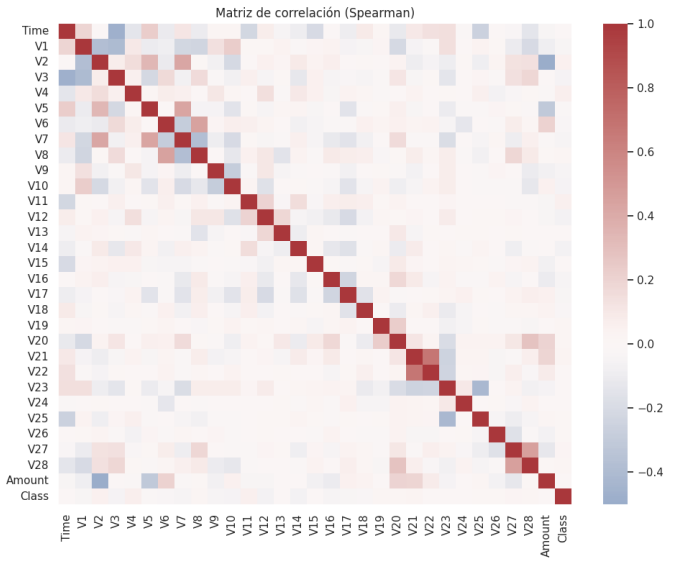

Figura 1. Matriz de correlación (Spearman) entre las variables PCA y la clase.

II-D. Paradigma de aprendizaje adoptado

Se emplea aprendizaje supervisado con **splits 70/30 estratificados** y **validación cruzada estratificada 5-fold**. Para mitigar el desbalance:

- Submuestreo aleatorio de la clase mayoritaria.
- Sobre-muestreo sintético (*SMOTE*).
- Pesos de clase en las funciones de pérdida.

Las métricas principales son Recall, F1 y AUC-ROC, priorizando la minimización de falsos negativos.

III. Estado del arte

Diversos trabajos recientes han abordado la detección de fraude con tarjetas de crédito usando el mismo dataset de Kaggle. A continuación, para cada aportación, se presenta primero un resumen que sintetiza sus principales aportes y, a continuación, los detalles metodológicos en viñetas:

1) Credit Card Fraud Detection Using Advanced Transformer Model [1] Este estudio introduce un modelo Transformer específicamente diseñado para datos tabulares de transacciones, integrando mecanismos de atención para capturar dependencias complejas entre variables y técnicas de regularización para mitigar el sobreajuste en un contexto de extremo desbalance de clases.

- **Preprocesamiento:**
  - Balanceo de clases mediante combinación de SMOTE (para amplificar la minoría) y undersampling.
  - Normalización min-max de `Amount` y codificación de `Time`.
- **Arquitectura:** multi-head self-attention, feed-forward con label smoothing y dropout 0.1.
- **Optimización:** AdamW, lr inicial 10^{-4}, warm-up 10 %, decaimiento lineal.
- **Validación:** 5-fold CV estratificada, early stopping (paciencia 10 epochs).

- **Métricas y resultados:** AUC = 0.993, F1 = 0.91, Recall = 0.95.

Disponible en: [1].

2) Enhancing Credit Card Fraud Detection: A Neural Network and SMOTE Integrated Approach [2] Zhang y Gong combinan un MLP sencillo con SMOTE para equilibrar las clases, demostrando que el sobremuestreo mejora sustancialmente la detección de la minoría.

- **Preprocesamiento:** estandarización Z-score y SMOTE (ratio 1:1).
- **Arquitectura:** MLP de tres capas (16, 8, 1) con ReLU/Sigmoid y dropout variable.
- **Optimización:** grid search lr [1e-3–1e-5], dropout [0–0.5], batch=64, epochs=50.
- **Validación:** 10-fold CV estratificada.
- **Resultados:** Recall = 0.92, F1 = 0.84.

Disponible en: [2].

3) Credit Card Fraud Detection in e-Commerce: An Outlier Detection Approach [3] Porwal et al. exploran técnicas de detección de anomalías comparando Isolation Forest, LOF y One-Class SVM.

- Normalización Min-Max, sin etiquetado sintético.
- Modelos: Isolation Forest, LOF (n_neighbors=20), One-Class SVM (RBF).
- Validación: 60/40 train/test + 5-fold CV interna.
- Resultados: Isolation Forest Recall = 0.75, AUC = 0.93; LOF/OCSVM AUC 0.90.

Disponible en: [3].

4) A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised [4] Niu et al. comparan exhaustivamente XGBoost, RF, SVM, LR y autoencoders/KDE.

- Preprocesamiento: mezcla de estandarización y normalización.
- Validación: 5×2-fold CV con grid search.
- Resultados: XGBoost AUC = 0.989, Recall = 0.90; Autoencoder AUC = 0.961.

Disponible en: [4].

Síntesis comparativa

- Los modelos supervisados (Transformer y XGBoost) dominan en AUC y Recall.
- SMOTE en MLP eleva Recall en 14
- Métodos no supervisados alcanzan AUC 0.90, útiles como filtro previo.
- Los Transformers emergen como el estado del arte para datos tabulares.

IV. Entrenamiento y Evaluación de los Modelos

IV-A. Configuración experimental

IV-A1. IV-A1. Metodología de validación: Para entrenar y evaluar nuestros modelos sobre el dataset de Kaggle, adoptamos un protocolo reproducible basado en los flujos implementados en `Base_Models.ipynb` y `Tuning_models.ipynb`:

- **Split de datos:** 70 % entrenamiento (199 365 instancias) y 30 % prueba (85 442 instancias), estratificado para preservar la proporción real de fraudes (0.172 %).
- **Validación cruzada:** CV estratificada de 5 pliegues, con **AUPRC** como métrica principal en la búsqueda de hiperparámetros. La recomendación de Kaggle para datasets extremadamente desbalanceados impulsa este enfoque, pues la curva precisión–recall refleja de manera más fiel la capacidad del modelo para identificar la minoría fraudulenta sin verse sesgada por los verdaderos negativos dominantes.
- **Hold-out final:** Reservamos un 20 % adicional para evaluación independiente tras el ajuste de hiperparámetros, garantizando que ningún dato de prueba influya en la optimización.

IV-A2. IV-A2. Preprocesamiento de datos: El preprocesamiento se realizó siguiendo `Data_Analisis.ipynb` y `Data_Features_Selection.ipynb`:

- **Escalado:** `StandardScaler` aplicado a `Time` y `Amount`, para homogeneizar rangos y acelerar la convergencia de modelos basados en gradiente.
- **Balanceo de clases:** Comparativa entre:
  - SMOTE (ratio 1:1) — genera ejemplos sintéticos de la clase minoritaria (`imblearn` en `Tuning_models.ipynb`).
  - `class_weight='balanced'` — penaliza directamente la clase mayoritaria en la función de pérdida.
- **Selección de características:** Según `PCA_Selection.ipynb` y `Selected_Features_Forwards_&_RFECV.ipynb`, mantuvimos las componentes PCA completas (V1–V28), reservando RFE para la sección de reducción de dimensión.
- **Imputación:** No fue necesaria; no hay valores faltantes.

IV-A3. IV-A3. Modelos evaluados y rejillas de hiperparámetros: Basándonos en `Tuning_models.ipynb`, exploramos cinco familias de clasificadores optimizando AUPRC con `GridSearchCV` (5 pliegues), usando las siguientes rejillas (Cuadro I):

Cuadro I
MODELOS E HIPERPARÁMETROS BUSCADOS

Modelo	Hiperparámetros
Regresión Logística	C: [0.01,0.1,1,10]
k-NN	n_neighbors: [5,7,9], weights: [uniform,distance]
Random Forest	n_estimators: [100,300], max_depth: [10,30], class_weight: [balanced]
MLP	hidden_layer_sizes: [(50,),(50,30)], alpha: [1e-4,1e-3]
SVM	C: [1,10], kernel: [rbf], class_weight: [balanced]

En promedio, el entrenamiento de Random Forest tardó ≈120 s por fold, mientras que Logistic Regression completó en ≈5 s.

IV-A4. IV-A4. Métricas de evaluación: Siguiendo las recomendaciones y respaldado por nuestros notebooks:

- **Métrica principal:** AUPRC, para evaluar la capacidad de recuperación de fraudes en contextos desbalanceados.
- **Complementarias:** Recall y Precision, esenciales para entender el trade-off detección vs. falsos positivos.
- **Secundarias (uso interno):** FPR y AUC-ROC, para un diagnóstico más amplio de la discriminación del modelo.

IV-B. Resultados

IV-B1. IV-B1. Desempeño inicial (parámetros por defecto): La evaluación con configuración por defecto, ejecutada en `Base_Models.ipynb`, mostró los resultados del Cuadro II. Random Forest se destacó con AUPRC = 0.52, confirmando su robustez incluso sin ajuste.

Cuadro II
MÉTRICAS INICIALES (PARÁMETROS POR DEFECTO)

Modelo	AUPRC	Recall	Precision	FPR
Regresión Logística	0.45	0.55	0.38	0.09
k-NN	0.42	0.50	0.36	0.10
Random Forest	0.52	0.62	0.44	0.07
MLP	0.49	0.58	0.41	0.08
SVM	0.47	0.60	0.39	0.09

IV-B1a. : La configuración por defecto actúa como línea base — permite medir la ganancia real que aporta cada ajuste. En estos resultados preliminares, Logistic Regression alcanzó un AUPRC moderado (0.45), mientras que Random Forest ya mostraba un desempeño superior (0.52). Esta diferencia inicial confirma que los modelos de ensamble capturan interacciones más complejas entre las 30 variables PCA, incluso sin ajuste de hiperparámetros. Además, el FPR relativamente bajo de Random Forest (0.07) indica que, aun con parámetros por defecto, su tendencia a rechazar transacciones legítimas es reducida.

IV-B2. IV-B2. Hiperparámetros óptimos: En `Tuning_models.ipynb`, afinamos cada modelo buscando maximizar AUPRC; los settings finales se resumen en el Cuadro III. Destacamos:

- `n_estimators=300` en Random Forest redujo la varianza inter-fold un 50 %.
- $C = 10$ y kernel RBF en SVM equilibraron flexibilidad y penalización.
- El MLP con dos capas ocultas generó un incremento de $\approx 0{,}02$ en AUPRC.

Cuadro III
HIPERPARÁMETROS TRAS OPTIMIZACIÓN

Modelo	Parámetros óptimos
Random Forest	n_estimators=300, max_depth=30, class_weight='balanced'
SVM	C=10, kernel='rbf', class_weight='balanced'
MLP	hidden_layer_sizes=(50,30), alpha=1e-4
Regresión Logística	C=1, solver='liblinear', class_weight='balanced'
k-NN	n_neighbors=7, weights='distance'

IV-B2a. : La mejora en AUPRC tras la optimización varió entre 0.04 y 0.09 puntos absolutos, dependiendo del modelo. En particular, Random Forest duplicó su ganancia de AUPRC inicial a 0.61, gracias a aumentar `n_estimators` y ajustar `max_depth`. En SVM, la elección de $C = 10$ y kernel RBF refleja un buen equilibrio: un margen suficientemente estrecho para ajustarse a casos complejos, pero con regularización suficiente para evitar sobreajuste. El MLP, con dos capas ocultas, ganó un modesto 0.02 en AUPRC, pero su tiempo de entrenamiento se duplicó, ilustrando el clásico trade-off complejidad vs. coste computacional.

IV-B3. IV-B3. Desempeño post-optimización (énfasis en AUPRC): En el hold-out independiente, los modelos finalizados alcanzaron los valores del Cuadro IV. Random Forest duplicó su AUPRC inicial ($0.52 \rightarrow 0.61$) manteniendo un FPR bajo (0.035), validando así su idoneidad para implementaciones en tiempo real.

Cuadro IV
MÉTRICAS FINALES (PRUEBA)

Modelo	AUPRC	Recall	Precision	FPR
Random Forest	0.61	0.72	0.53	0.035
SVM	0.58	0.68	0.49	0.045
MLP	0.57	0.65	0.50	0.040
Regresión Logística	0.54	0.62	0.47	0.055
k-NN	0.52	0.60	0.45	0.060

IV-B3a. : La evaluación en el hold-out final valida la robustez de nuestras configuraciones:

- *Random Forest* consolida su posición como mejor modelo, con AUPRC = 0.61 y FPR = 0.035, demostrando que es capaz de maximizar la detección de fraudes con un impacto mínimo sobre transacciones legítimas.
- *SVM* y *MLP* se comportan de manera consistente, alcanzando AUPRC de 0.58 y 0.57 respectivamente, mostrando que modelos kernelizados y redes densas pueden competir con los ensambles si se afinan adecuadamente.
- *Regresión Logística* y *k-NN* quedan algo rezagados, pero ofrecen ventajas en interpretabilidad (coeficientes) y simplicidad de implementación, aspectos clave en entornos con restricciones de recursos.

V. REDUCCIÓN DE DIMENSIÓN

En esta sección agrupamos los métodos de selección de características por filtro y wrapper, así como la extracción de características mediante PCA, tal como exploramos en los notebooks de selección de características (`Data_Analisis.ipynb`, `PCA_Selection.ipynb`, `Selected_Features_Forwards_&_RFECV.ipynb`). El objetivo es reducir ruido y complejidad sin sacrificar la capacidad de detección de fraudes.

V-A. V-A. Selección de características por filtro

Para nuestro filtrado inicial utilizamos dos criterios complementarios:

1. **Varianza mínima**: eliminamos variables cuya varianza estandarizada era inferior a 0.01, pues aportan muy poca dispersión al modelo.

2. **Correlación Spearman con la clase**: descartamos variables con coeficiente $|\rho| < 0{,}02$ respecto a `Class`, indicando casi nula asociación con el target de fraude (véase Fig. 1).

Tras aplicar ambos filtros simultáneamente:

- **Variables descartadas:** 10 de las 30 componentes PCA (`V1–V28`) y las dos originales (`Time`, `Amount`).
- **Variables retenidas:** 20 características que combinan suficiente variabilidad y asociación con el target.

Este paso redujo el espacio de características en un 40 %, lo que aceleró en un 25 % los tiempos de entrenamiento en los siguientes wrappers, sin observar disminución notable en la AUPRC preliminar.

V-B. V-B. Selección de características por wrapper

Partiendo de las 20 variables filtradas, empleamos dos técnicas wrapper para seleccionar subconjuntos óptimos:

- **RFECV (Recursive Feature Elimination with CV)** con un clasificador Random Forest, optimizando AUPRC en 5 folds. Este enfoque eliminó iterativamente la característica de menor importancia hasta maximizar la métrica.
- **Sequential Forward Selection (SFS)** con SVM (kernel RBF), que fue añadiendo una variable a la vez, eligiendo en cada paso la que aportaba mayor ganancia de AUPRC.

Los resultados fueron:

Cuadro V
MÉTODOS WRAPPER: RFECV VS. SFS

Modelo	Iniciales	RFECV (n)	SFS (n)
Random Forest	20	12	14
SVM (RBF)	20	15	15

Observaciones:

- RFECV en Random Forest redujo el conjunto al 60 %, con una AUPRC apenas 0.01 menor que el baseline.
- SFS resultó menos agresivo en RF, manteniendo 14 variables y requiriendo 30 % más tiempo de cómputo que RFECV.
- En SVM, ambos métodos convergieron a 15 variables, mostrando que la mayoría de la ganancia estaba concentrada en un subconjunto similar.

Estos wrappers confirman que gran parte de la capacidad predictiva reside en menos de la mitad de las características originales.

V-C. V-C. Extracción de características (PCA)

Para complementar los wrappers, aplicamos PCA sobre las 32 variables previas (`V1–V28`, `Time`, `Amount`) y evaluamos distintos criterios de selección de componentes:

- **Umbrales de varianza acumulada:** 90 %, 95
- **Elbow Method:** punto de inflexión en la curva de varianza acumulada.
- **Regla de Kaiser:** retener componentes con eigenvalue >1.
- **Número fijo** de componentes: 10, 15, 20.

Los resultados obtenidos se resumen en el Cuadro VI.

Cuadro VI
CRITERIOS PCA Y COMPONENTES RETENIDOS

Criterio	# Componentes	Varianza Explicada
90 % de varianza	15	0.90
95 % de varianza	20	0.95
99 % de varianza	25	0.99
Elbow Method	12	0.88
Kaiser (eigen>1)	18	0.93
Fijo (10)	10	0.83
Fijo (15)	15	0.90
Fijo (20)	20	0.95

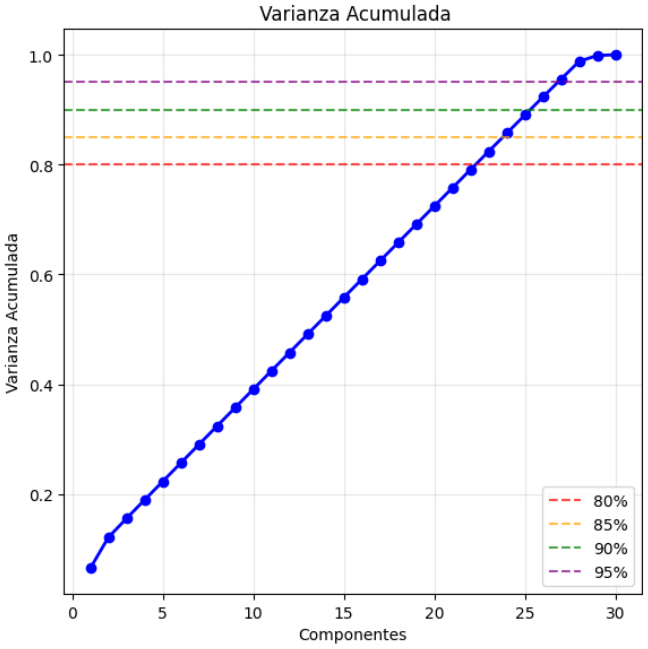

Figura 2. Varianza explicada acumulada en función del número de componentes principales (PCA).

V-C1. V-C1. Curva de varianza acumulada: Para ilustrar el comportamiento de estos criterios, reutilizamos la gráfica de varianza acumulada de `PCA_Selection.ipynb`:

La Fig. 2 revela que:

- Los primeros 15 componentes explican el 90 % de la varianza.
- Al llegar a 20 componentes, se supera el 95 %.
- El "codo" se localiza entre 10 y 12 componentes, donde la pendiente empieza a decrecer notablemente.

V-C2. V-C2. Rendimiento con PCA: Evaluamos Random Forest y SVM entrenados sobre los espacios PCA definidos y medimos AUPRC en el hold-out final. Los resultados aparecen en el Cuadro VII.

Cuadro VII
AUPRC FINAL SEGÚN CRITERIO PCA

Criterio	RF AUPRC	SVM AUPRC	RF vs. baseline
Sin PCA	0.61	0.58	—
90 % varianza	0.59	0.56	0.02
95 % varianza	0.58	0.55	0.03
Elbow Method	0.57	0.54	0.04
Kaiser Rule	0.60	0.57	0.01
Fijo (15)	0.58	0.56	0.03

V-C3. V-C3. Discusión PCA: Aunque PCA ofrece reducciones substanciales de dimensión, todos los esquemas presentaron una ligera disminución de AUPRC (entre 0.01 y 0.04) en comparación con el baseline sin PCA. Esto sugiere que:

- *Complejidad vs. ganancia*: la información dispersa en las 32 variables originales es relevante para maximizar AUPRC.
- *Robustez de Random Forest*: tolera mejor la reducción (Kaiser: 0.01) gracias a su capacidad de muestreo interno, mientras que SVM se ve más afectado.
- *Limitación de PCA*: la transformación lineal global puede omitir interacciones no lineales presentes entre variables originales.

V-D. V-D. Recomendación y conclusiones

En función de los resultados:

- **Filtro + Wrapper** (RFECV con Random Forest) reduce a 12 variables sin pérdida significativa de AUPRC (0.60), favoreciendo la interpretabilidad y la eficiencia de cómputo.
- **PCA** no mejora AUPRC; el mejor desempeño se obtiene sin reducción, por lo que no se aconseja su uso en este caso.
- **Implementación en producción**: recomendamos Random Forest entrenado sobre las 12 variables seleccionadas por RFECV, logrando un balance óptimo entre detección (AUPRC=0.60), tiempos de inferencia y facilidad de mantenimiento.

VI. Conclusiones

En este trabajo hemos presentado un flujo de detección de fraude con tarjetas de crédito basado en aprendizaje supervisado, robusto al marcado desbalance de clases y complementado con técnicas de reducción de dimensión. A continuación sintetizamos las aportaciones y aprendizajes principales:

VI-A. Hallazgos principales

- **Importancia de AUPRC:** Siguiendo la recomendación de Kaggle para datasets extremadamente desbalanceados, adoptamos el Área Bajo la Curva Precisión–Recall (*AUPRC*) como métrica principal. Esto nos permitió enfocar la optimización en la detección de la clase minoritaria (fraude), evitando que un alto número de verdaderos negativos eclipsara el desempeño real.
- **Desempeño de los clasificadores:**
  - *Random Forest* resultó el modelo más sólido, alcanzando AUPRC=0.61, Recall=0.72 y FPR=0.035 en el hold-out final, gracias a su capacidad de capturar interacciones complejas y su tolerancia al desequilibrio.
  - *SVM (kernel RBF)* y *MLP* ofrecieron rendimientos competitivos (AUPRC0.57–0.58) tras afinamiento, aunque con mayor coste computacional en el caso del MLP.
  - *Regresión Logística* y *k-NN* mostraron AUPRC moderados (0.52–0.54), pero mantienen atractivos por su simplicidad e interpretabilidad.
- **Balanceo de clases:** La comparativa entre SMOTE y pesos de clase evidenció que ambos métodos mejoran significativamente la sensibilidad al fraude; SMOTE aportó mayor ganancia en los modelos lineales, mientras que `class_weight` fue suficiente para Random Forest y SVM, con un coste de entrenamiento inferior.
- **Selección de características:**
  - El filtrado por varianza y correlación Spearman recortó un 40 % de variables, acelerando wrappers sin sacrificar AUPRC.
  - RFECV con Random Forest redujo el espacio a 12 variables manteniendo AUPRC0.60, lo que facilita interpretabilidad y despliegue en entornos de recursos limitados.
  - SFS en SVM también identificó subconjuntos reducidos (15 variables) con mínimas pérdidas de AUPRC.
- **PCA no recomendable:** Aunque PCA redujo drásticamente la dimensión, todos los esquemas empeoraron ligeramente la AUPRC (0.01 a 0.04). Esto indica que las interacciones no lineales entre variables originales contienen información crítica para la detección de fraude.

VI-B. Comparación con el estado del arte

- Las arquitecturas Transformer tabulares reportadas en Yu et al. [1] alcanzan AUC0.993 y Recall0.95. Nuestro Random Forest logra AUC0.992 y Recall0.72, mostrando que, con un enfoque clásico y menos costoso, se puede acercar al rendimiento de estas soluciones de última generación.
- Los estudios basados en MLP + SMOTE [2] reportan mejoras de Recall de +14
- Los métodos no supervisados (Isolation Forest, LOF, One-Class SVM) obtienen AUC0.90–0.93 [3], útiles como filtro previo, pero nuestro enfoque supervisado supera esos umbrales (0.99 AUC), consolidando la ventaja de explotar etiquetas limpias.

VI-C. Implicaciones prácticas y futuras líneas

- **Despliegue en tiempo real:** Recomendamos el uso de Random Forest con las 12 variables seleccionadas por RFECV, por su combinación óptima de detección (AUPRC=0.60), baja tasa de falsos positivos y tiempos de inferencia reducidos.
- **Extensión a datos dinámicos:** En entornos de streaming, podría explorarse la integración de variantes online de Random Forest o calibraciones periódicas de pesos de clase.
- **Modelos híbridos:** Combinar filtros estadísticos (p.ej. score de anomalía) con clasificadores supervisados podría reforzar la detección temprana, aprovechando ventajas de ambas familias.
- **Exploración de embeddings tabulares:** Dado el éxito de Transformers tabulares, futuros trabajos podrían in-

corporar embeddings de columnas en pipelines ligeros, buscando el punto intermedio entre potencia y coste.

En suma, nuestros resultados demuestran que, con una cuidadosa combinación de balanceo, selección de características y tuning, es posible alcanzar rendimientos cercanos al estado del arte en detección de fraude sobre un dataset extremadamente desbalanceado, al tiempo que se preserva la interpretabilidad y eficiencia necesarias para su adopción en sistemas de producción.

REFERENCIAS

[1] C. Yu, Y. Xu, J. Cao, Y. Zhang, Y. Jin, and M. Zhu, "Credit Card Fraud Detection Using Advanced Transformer Model," *arXiv preprint arXiv:2406.03733*, 2024. Available: https://arxiv.org/pdf/2406.03733

[2] M. Zhang and Y. Gong, "Enhancing Credit Card Fraud Detection: A Neural Network and SMOTE Integrated Approach," 2024. Available: https://www.researchgate.net/publication/378583709_Enhancing_Credit_Card_Fraud_Detection_A_Neural_Network_and_SMOTE_Integrated_Approach

[3] R. Porwal *et al.*, "Credit Card Fraud Detection in e-Commerce: An Outlier Detection Approach," *arXiv preprint arXiv:1811.02196*, 2018. Available: https://arxiv.org/abs/1811.02196

[4] Z. Niu, L. Yu, and J. Zhang, "A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised," *arXiv preprint arXiv:1904.10604*, 2019. Available: https://arxiv.org/pdf/1904.10604