**Trabajo Práctico Grupal Minería de Datos**

**Sistemas Inteligentes**

A colorful crest with a bird and a dolphin

AI-generated content may be incorrect.

Funes Norman

Fruhwirth Juan

Francone Nicolás

Gallo Kuhn Mauro Andrés

Fernandez Calvo Santiago Gastón

2025.

Universidad del Salvador

Facultad de Ingeniería

**Contenido**

[Objetivo 3](#_Toc201890541)

[Dataset 3](#_Toc201890542)

[Link 3](#_Toc201890543)

[Descripción 3](#_Toc201890544)

[Cantidad de Ejemplos y de Atributos 4](#_Toc201890545)

[Tareas 4](#_Toc201890546)

[Limpieza 4](#_Toc201890547)

[Selección 6](#_Toc201890548)

[Integración 7](#_Toc201890549)

[Transformación 7](#_Toc201890550)

[Minería 8](#_Toc201890551)

[Evaluación 9](#_Toc201890552)

[Conclusiones 11](#_Toc201890553)

# Objetivo

El objetivo de este trabajo práctico es aplicar el proceso de Descubrimiento de Conocimiento en Datos (KDD) sobre un dataset real, utilizando técnicas de aprendizaje no supervisado con detección de anomalías. El propósito principal es detectar posibles transacciones fraudulentas en tarjetas de crédito, sin utilizar etiquetas durante el entrenamiento, simulando un escenario donde las instancias anómalas no se conocen de antemano.

Para ello se utilizó Python junto a la biblioteca scikit-learn, trabajando en un entorno Jupyter Notebook que permitió implementar cada fase del proceso, incluyendo análisis, estandarización, detección de anomalías y evaluación, en base a las métricas obtenidas y la visualización de los resultados.

# Dataset

## Link

[Credit Card Fraud Detection - Kaggle](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

## Descripción

El dataset utilizado contiene transacciones con tarjetas de crédito realizadas por usuarios europeos durante dos días de septiembre de 2013. Está especialmente diseñado para el análisis de fraudes, ya que incluye tanto operaciones legítimas como fraudulentas, aunque estas últimas representan un porcentaje muy bajo del total.

Todas las variables son numéricas. La mayoría (V1 a V28) surgen de una transformación PCA aplicada por motivos de confidencialidad. Solo dos columnas conservan sus valores originales:

* Time, que representa los segundos desde la primera transacción registrada.
* Amount, que indica el monto de la transacción.

Esto nos adelanta que de las 31 columnas del dataset, solo 3 de ellas no se encuentran a la misma escala que las otras restantes.

El atributo Class indica si la transacción fue fraudulenta (1) o no (0), y fue utilizado únicamente en la fase de evaluación, ya que el entrenamiento del modelo fue no supervisado.

## Cantidad de Ejemplos y de Atributos

* Total de ejemplos: 284.807
* Total de atributos: 31
* 30 Atributos predictivos (V1 a V28, Time y Amount).
* 1 Atributo clasificador (Class, solo usado para evaluación).

# Tareas

## Limpieza

Se realizó una inspección inicial del dataset para poder identificar posibles valores nulos, duplicados, tipos de dato de cada columna, entre otras.

A través de este análisis, se identificó que todas las columnas contaban con datos de tipo float y uno de tipo int.

A screenshot of a computer

AI-generated content may be incorrect.

También, se verificó la distribución del dataset, determinando que solo el 0,17% de las transacciones corresponde a fraudes (1), siendo el 99,83% restante transacciones normales (0). Este desbalance significativo influyó mucho en las decisiones tomadas en etapas posteriores.

A blue rectangular bar graph

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

Por otro lado, se confirmó que no existen valores nulos en ninguno de los atributos del dataset, pero sí la presencia de 1081 filas duplicadas. Sin embargo, debido a que varios ejemplos de fraude entraban en esta categoría, eliminarlos iba a implicar quitar una parte importante de los pocos ejemplos valiosos con los que el dataset contaba.

## Selección

En esta etapa se analizaron los atributos disponibles para decidir cuáles conservar en el proceso de modelado. Si bien originalmente se pensó en eliminar la variable Time (que representa los segundos transcurridos desde la primera transacción registrada) por su formato acumulativo y limitada interpretabilidad, finalmente se optó por transformarla en una nueva variable llamada Hour, que indica la hora del día en que ocurrió la transacción.

Esta transformación permitió conservar el valor temporal de manera más útil, ya que algunos fraudes pueden concentrarse en ciertos momentos del día (por ejemplo, durante la madrugada), lo cual puede ser capturado por el modelo como comportamiento anómalo.

El resto de los atributos fueron considerados relevantes. Como se explicó anteriormente, las variables V1 a V28 fueron generadas mediante una transformación PCA previa, lo que impide aplicar criterios de selección semántica, ya que no tienen significado explícito. Sin embargo, al conservar la variabilidad original, se decidió mantenerlas todas.

## Integración

Esta etapa no fue necesaria, ya que el análisis se basa en un único dataset que contiene todas las variables requeridas.

No fue necesario integrar múltiples fuentes de información ni realizar uniones entre tablas, ya que los datos provienen de un único archivo consolidado (creditcard.csv), preparado previamente para tareas de detección de fraude con aprendizaje automático.

## Transformación

En esta etapa se aplicaron varias transformaciones clave para mejorar la calidad del dataset antes de entrenar los modelos. Primero se generó un nuevo atributo llamado Hour, derivado del atributo Time, que representa la hora del día en que ocurrió la transacción. Esto se hizo dividiendo el tiempo (en segundos) por 3600 y aplicando módulo 24, lo que permite capturar patrones temporales que podrían estar relacionados con comportamientos fraudulentos.

El atributo Amount fue transformado aplicando la función log1p para reducir su asimetría. Este atributo presenta una distribución muy sesgada, con muchos valores pequeños y algunos muy altos. La transformación logarítmica comprime los valores extremos sin perder proporcionalidad, mejorando el rendimiento de los algoritmos de detección de anomalías.

A graph of two people

AI-generated content may be incorrect.

Finalmente, se escalaron todos los atributos del dataset (incluyendo Amount, Hour y V1–V28) utilizando StandardScaler. Aunque los atributos del PCA ya estaban centrados cerca de cero, no se podía garantizar que estuvieran en la misma escala que las nuevas variables. Escalar todo en conjunto aseguró que ninguna variable domine el comportamiento del modelo debido a diferencias de magnitud.

Además, no se aplicaron otras transformaciones porque:

* El dataset no incluye variables categóricas.
* Eliminar outliers no es recomendable en este caso, ya que los fraudes suelen representar justamente valores extremos que se desean detectar.
* Ya se aplicó una reducción de dimensionalidad (PCA) durante la construcción original del dataset.

## Minería

Para entrenar y evaluar el modelo se aplicó validación cruzada con KFold (5 particiones), garantizando que cada fold contuviera una muestra distinta de los datos. En cada iteración, se entrenó sobre 4 folds y se evaluó sobre el restante. Esta estrategia permitió obtener una evaluación más robusta del desempeño general del modelo.

El modelo seleccionado fue **Isolation Forest**, una técnica de aprendizaje no supervisado diseñada para la detección de anomalías. Este algoritmo construye múltiples árboles aleatorios y considera como anomalías a aquellas instancias que se aíslan rápidamente durante la partición de los datos. Fue elegido por su eficacia en grandes volúmenes de datos y por su capacidad para identificar patrones anómalos sin requerir etiquetas durante el entrenamiento.

Se configuró con el parámetro contamination=0.01, lo que indica que el modelo asume que aproximadamente un 1% de las transacciones son anómalas, valor razonable considerando la proporción real de fraudes en el dataset.

Luego de aplicar una primera versión del modelo con hiperparámetros por defecto, se decidió realizar una segunda iteración incorporando una búsqueda de hiperparámetros con Grid Search manual. El objetivo fue encontrar una combinación que mejorara el rendimiento general del modelo, especialmente la métrica F1 Score, que refleja el balance entre precision y recall.

Como GridSearchCV no es aplicable directamente a problemas no supervisados (porque requiere etiquetas durante el entrenamiento), se implementó una versión manual: se probaron múltiples combinaciones de hiperparámetros (como cantidad de árboles, proporción de muestras y tasa de anomalías) mediante validación cruzada con KFold, y se eligió la combinación con mejor F1 promedio en los 5 folds.

Esta búsqueda permitió ajustar el comportamiento del modelo sin violar el enfoque no supervisado, ya que las etiquetas reales solo se usaron en la fase de evaluación.

Adicionalmente, se exploró el uso del modelo Local Outlier Factor (LOF), basado en la densidad local de los puntos. Sin embargo, debido al gran volumen de datos y la alta dimensionalidad del dataset, el modelo resultó muy lento y con métricas de rendimiento cercanas a cero. Por esta razón, fue descartado y no se utilizó en la solución final.

## Evaluación

Dado el fuerte desbalance de clases, no se utilizó accuracy como métrica principal, ya que sería engañosa (un modelo que predice siempre “no fraude” tendría alta exactitud, pero no detectaría nada útil).

En su lugar, se utilizaron las siguientes métricas específicas:

* Precision
* Recall
* F1 Score

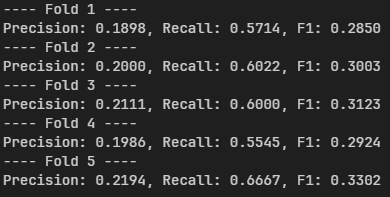
Estas métricas se calcularon en cada fold y luego se compararon globalmente para evaluar la estabilidad y efectividad del modelo.

A black screen with white text

AI-generated content may be incorrect.

En la primera iteración, el modelo logró buenos niveles de recall, lo que significa que fue capaz de detectar muchos de los fraudes reales. Sin embargo, la precision fue baja (alrededor del 10%), lo cual es esperable en un problema con tan pocos casos positivos. Los valores de F1 Score fueron aceptables, pero existía margen de mejora.

Tras realizar la búsqueda de hiperparámetros mencionada en la sección anterior, los resultados mejoraron considerablemente: la precision se duplicó (pasando de ~10% a ~20%), el F1 Score se incrementó de manera significativa (de ~0.18 a ~0.30), y el recall se mantuvo relativamente alto, apenas por debajo del valor original.



Esta nueva configuración permitió un modelo más equilibrado: detecta una gran proporción de fraudes (recall elevado) pero con menor cantidad de falsos positivos, mejorando su utilidad en un escenario real.

En el siguiente gráfico se puede ver cómo variaron las métricas en cada fold:

A graph with lines and dots

AI-generated content may be incorrect.

De la misma forma, se generó una visualización con PCA para representar cómo se distribuyen las transacciones normales y las detectadas como anómalas:

A graph with red and blue dots

AI-generated content may be incorrect.

# Conclusiones

El trabajo permitió aplicar un proceso completo de KDD sobre un dataset real, enfocándose en la detección de fraudes con técnicas no supervisadas. Se trabajó con datos limpios y preprocesados, y se utilizó Isolation Forest como modelo principal por su capacidad para detectar patrones anómalos sin necesidad de etiquetas.

A pesar del fuerte desbalance, el modelo logró detectar una parte importante de los fraudes, con niveles aceptables de *recall* y *F1 score*. Si bien la *precision* fue baja, esto era esperable por la naturaleza del problema. La cross-validation mostró resultados estables entre folds.

En resumen, se logró cumplir el objetivo del trabajo: detectar anomalías en transacciones con un enfoque práctico, reproducible y sin uso de etiquetas durante el entrenamiento. La mejora lograda a través de la segunda iteración con búsqueda de hiperparámetros demuestra que incluso en problemas no supervisados, ajustar el modelo puede generar resultados más equilibrados y útiles. El modelo final mantuvo un buen recall mientras duplicaba la precision, alcanzando un F1 Score mucho más sólido que en la primera versión.