PROYECTO GRAFOS
DETECCIÓN DE FRAUDE EN TRANSACCIONES DE TARJETA DE CRÉDITO

Daniela Parra[a,c], Manuel Santiago Hernández Ayala [a,c], Paula Jaimes [a,c],
Lina Varón[a,c] , Jenny Ortiz [a,c]
Sergio Alberto Mora[b,c]

[a]*Estudiantes de Maestría en Analítica para la Inteligencia de Negocios*
[b]*Profesor, Departamento de Ingeniería Industrial*
[c]*Pontificia Universidad Javeriana, Bogotá, Colombia*

ENTENDIMIENTO DEL NEGOCIO

Background

Según el Reporte Anual de Tendencias de Fraude Omnicanal publicado en 2023 por la red de Inteligencia Global de Transunion, en la cual se toma información de 18 países, entre ellos Estados Unidos y Colombia, el fraude digital volvió a niveles experimentados antes de la pandemia, impulsado en gran medida por el crecimiento de un 80% en las transacciones digitales. Así mismo, el reporte destaca un aumento en la sofisticación de las actividades fraudulentas, particularmente en el robo de identidades y las estafas relacionadas con la identidad (phishing, vishing, smishing).

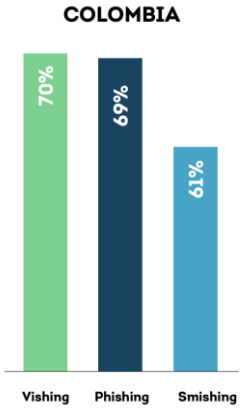


Figura 1. Principales preocupaciones relacionadas con el fraude en Colombia

En este reporte, también encontramos algunas cifras clave que nos ayudan a contextualizar la dimensión del problema, el fraude con identidades sintéticas aumentó un 76% desde 2019. Este tipo de fraude, que implica la creación de identidades falsas con información real y falsa, se ha convertido en una de las principales preocupaciones; el saldo atribuible a posibles identidades sintéticas en solicitudes de crédito en EE.UU. alcanzó los 4.600 millones de dólares en 2022. Esta cifra, la más alta de la historia, representa un crecimiento del 27% desde 2020.

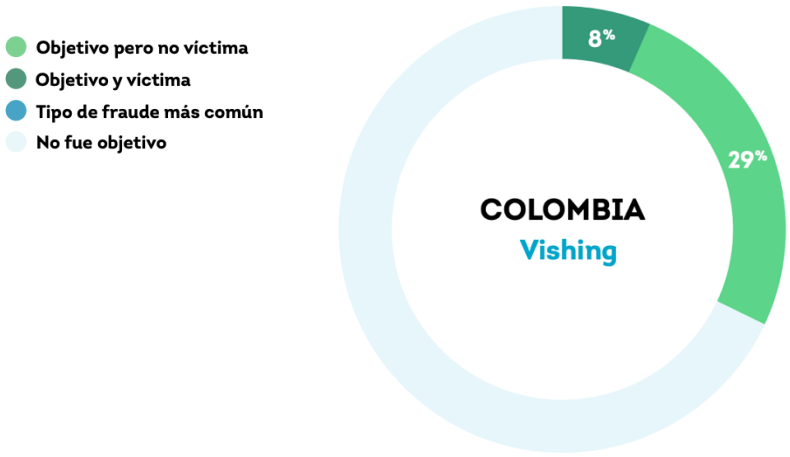


Figura 2. Denuncias por fraude entre septiembre y diciembre de 2022

El 52% de los consumidores a nivel global afirman haber sido blanco de un intento de fraude entre septiembre y diciembre de 2022, las industrias con mayor impacto han sido viajes, logística y servicios financieros. El impacto financiero del fraude afecta enormemente tanto a las instituciones como a los consumidores, y el costo asociado con el fraude se extiende más allá de las pérdidas directas e incluye costos operativos y de reputación. En Colombia por ejemplo, el 51% de los consumidores colombianos ha abandonado una solicitud online o un formulario de productos financieros debido a la falta de confianza en la protección de sus datos (TransUnion 2023 Informe Sobre El Estado Del Fraude Omnicanal, n.d.). Como vemos en la siguiente figura, dada la importancia de la seguridad para los usuarios, detectar el fraude y prevenirlo es fundamental para las empresas que ofrecen servicios en línea.

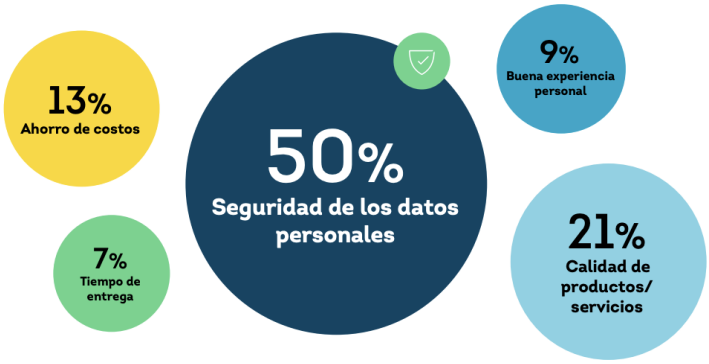


Figura 3. Cualidades preferidas por los consumidores en las empresas online.

En este contexto, el uso de grafos ha emergido como una herramienta poderosa para abordar problemas de detección de fraude. Los grafos permiten modelar las relaciones entre diferentes entidades, como usuarios, transacciones, dispositivos y ubicaciones, proporcionando una visión más completa de las interacciones en un sistema. En particular, los grafos son útiles para identificar patrones complejos de comportamiento fraudulento que pueden ser difíciles de detectar utilizando métodos tradicionales. Por ejemplo, las transacciones fraudulentas a menudo están vinculadas a comportamientos atípicos que pueden ser más evidentes cuando se visualizan en un grafo.

La técnica node2vec, que permite generar embeddings de nodos en un grafo, ha demostrado ser especialmente eficaz para capturar estas relaciones al transformar las conexiones en un espacio vectorial denso, los modelos pueden identificar más fácilmente las transacciones que siguen patrones sospechosos. Dicho esto, en este documento se explorará la utilización de estas técnicas para identificar transacciones fraudulentas en un conjunto de datos de transacciones con tarjetas de crédito.

Business goal

Realizar una clasificación de las transacciones legítimas y fraudulentas de tarjetas de crédito de 983 clientes y 693 comercios durante el período del 1 de enero de 2019 al 31 de diciembre de 2020.

Data mining goal

Realizar una clasificación binaria mediante la implementación grafos que permita separar las transacciones legítimas de las transacciones fraudulentas.

Data mining success criterio

Obtener un AUC del 95% y accuracy de 90% en la clasificación de transacciones fraudulentas

Entendimiento de los datos

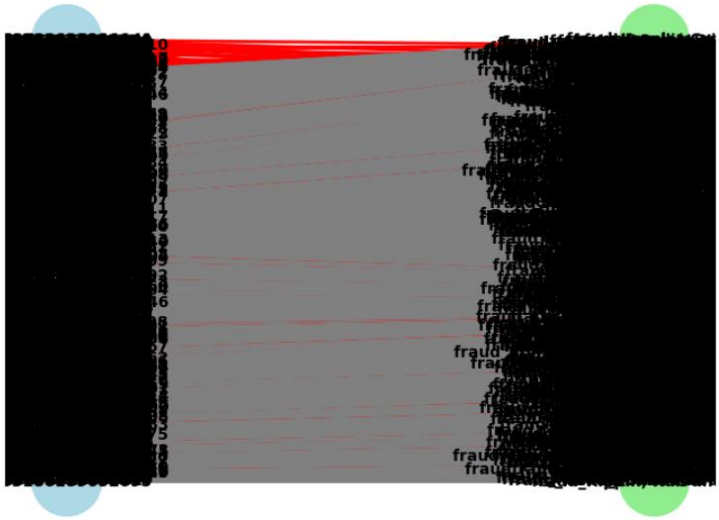
Para la construcción del grafo, se usa base de datos de Kaggle (https://www.kaggle.com/datasets/kartik2112/fraud-detection) que contiene información de 1289169 transacciones legítimas y 7506 fraudulentas realizadas con tarjeta de crédito. Esta base consta de un total de 20 variables, de las cuales 4 son usadas para la construcción del grafo:

cc_num	Id del usuario que realiza la compra	Nodo origen
merchant	Id del comercio donde se realiza la compra	Nodo destino
amt	Monto de la transacción	Peso de la arista
Is_fraud	Bander que toma el valor de 1 si se trata de una transacción fraudulenta.	Etiqueta de la arista

Las demás variables están relacionadas con la fecha de la transacción, la categoría de compra y la información sobre el lugar donde se realiza la transacción. Estas variables se incorporarán durante el entrenamiento del modelo para integrarlas con los embeddings generados a partir del grafo.

El grafo construido a partir de las cuatro variables mencionadas es un tipo de grafo bipartito dado que los nodos que conforman las aristas se dividen en dos conjuntos disjuntos: nodos de usuarios y nodos de comercios. La figura 1 es una representación gráfica del grafo bipartito, donde los nodos azules de la izquierda representan los usuarios que realizan la compra, los nodos verdes representan los comercios, las aristas en gris representan las transacciones legítimas y las aristas rojas representan las transacciones fraudulentas.

Figura 4. Grafo bipartito de transacciones

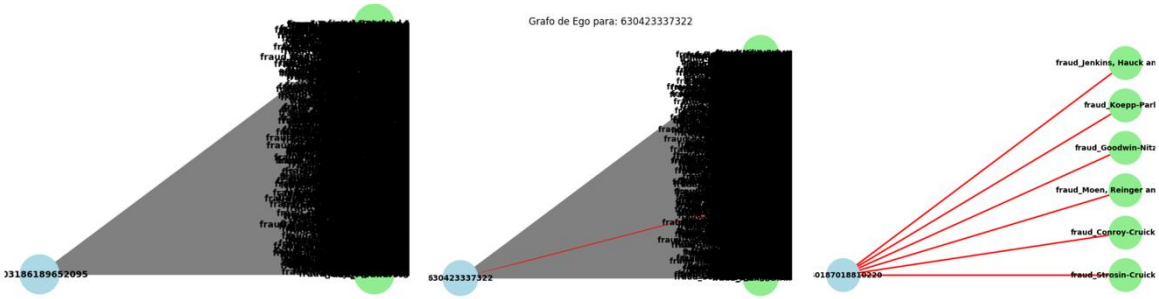


Fuente: Elaboración propia

El grafo consta de 1.676 nodos, de los cuales 983 corresponden a usuarios y 693 a comercios. Teniendo en cuenta que en el grafo bipartito solo se considera una arista entre cada par único de nodos, el grafo presenta un total de 479.072 aristas, de las cuales 3.730 están etiquetadas como fraude. Este desbalance en la distribución de conexiones fraudulentas sugiere la necesidad de realizar más adelante un proceso de re-muestreo para optimizar los resultados del modelo entrenado.

En la figura 2 se observan tres ejemplos de grafos de ego para nodos de usuarios distintos. Se puede observar que cada nodo usuario está conectado a múltiples comercios. En el primer caso, el usuario tiene conexión a una alta cantidad de comercios y no presenta transacciones fraudulentas. El segundo caso corresponde a un usuario con comportamiento de compra parecido, sin embargo, en su historial de compra se reporta una transacción fraudulenta. El tercer caso corresponde a un usuario cuyo historial de compras está asociado a sólo 6 conexiones con comercios y todas las transacciones han sido fraudulentas.

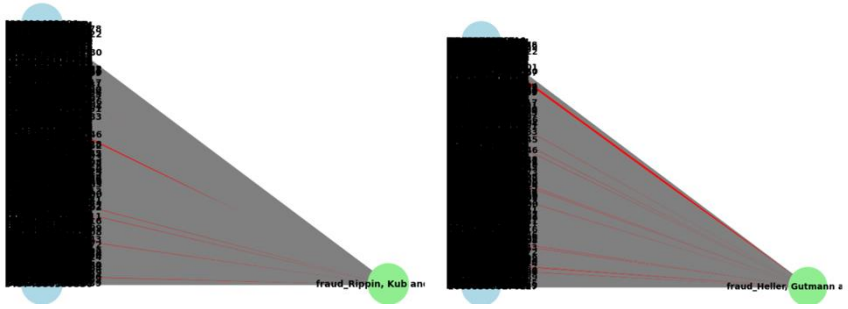
Figura 5. Grafos de ego de nodos de usuarios.



Fuente: Elaboración propia

Por otro lado, en la figura 3 se presentan dos ejemplos de grafos de ego de comercios, que han recibido transacciones de un gran número de usuarios, algunas de las cuales han sido clasificadas como fraude.

Figura 6. Grafos de egos de nodos de comercio.

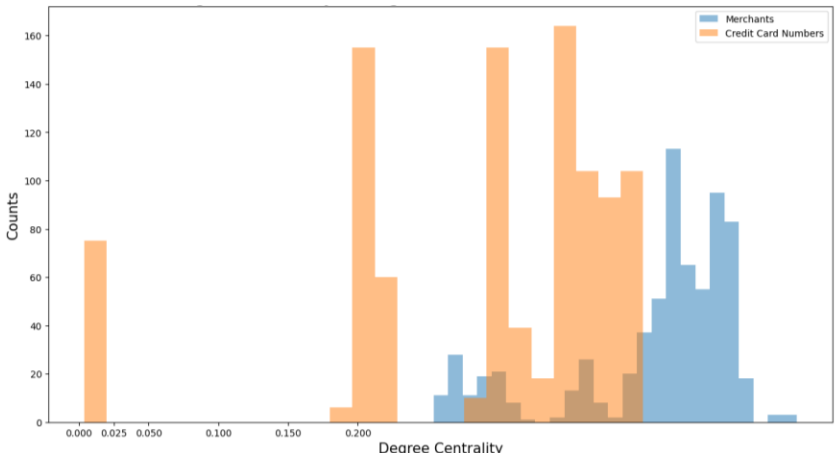


Fuente: Elaboración propia

Adicionalmente, se realiza un análisis para conocer el grado de centralidad de los usuarios y comercios de la red analizada. En la figura 4, se observa que la mayor parte de comercios tienen un grado de centralidad entre 40% y 50%, es decir, han recibido transacciones de aproximadamente la mitad de todos los usuarios en la red, lo que sugiere que en general los establecimientos son populares y relevantes para los consumidores. Los usuarios por su parte presentan menor grado centralidad (en su mayoría entre 20% y 40%) y existe un número significativo de usuarios con una utilización muy baja de sus tarjetas de crédito (grados de centralidad menores a 2.5 %).

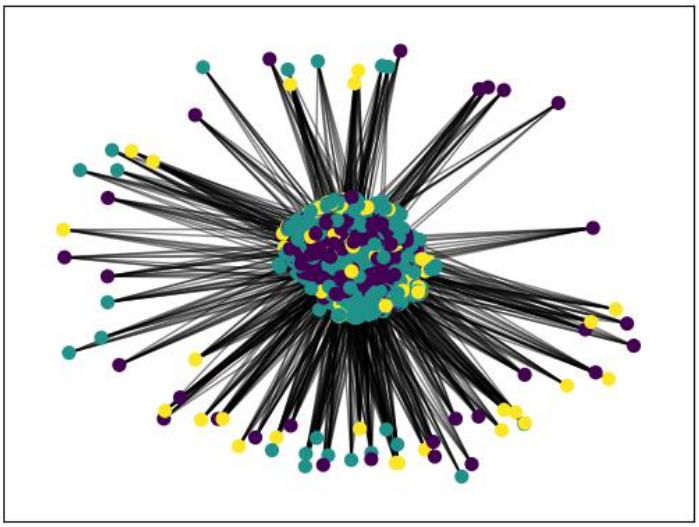
Por último, utilizando el método de Louvain, se realiza detección de comunidades dentro de los dos conjuntos (Usuarios y comercios). Estas comunidades, representadas en el gráfico 5, pueden consistir en grupos de personas que tienden a transaccionar con los mismos comercios o grupos de comercios que son frecuentados por las mismas personas. El método aplicado identifica la existencia de tres comunidades. La comunidad con la mayor cantidad de transacciones fraudulentas es la ‘1’ con un número de transacciones fraudulentas de 3.606.

Figura 7. Histograma de grados de centralidad de usuarios y comercios.



Fuente: Elaboración propia

Figura 8. Comunidades identificadas en el grafo.



	is_fraud
0	2855
1	3606
2	1045

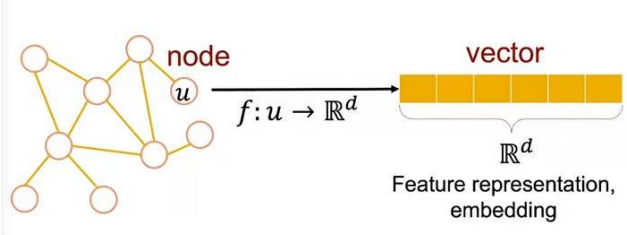
Fuente: Elaboración propia

Preparación de los datos

Para la preparación de los datos, se realiza la conversión al formato compatible con PyTorch Geometric. En este caso, el grafo bipartito se transforma al formato Data de PyTorch Geometric utilizando networkx, lo que facilita el procesamiento y el cálculo de embeddings en un entorno compatible con PyTorch.

La configuración de los embeddings, que son representaciones numéricas de los nodos en un espacio de menor dimensión, se utiliza el modelo Node2Vec de PyTorch Geometric. Este modelo se configuró con los siguientes parámetros clave: embedding_dim=128, walk_length=20, context_size=10,

walks_per_node=10, num_negative_samples=1, y los hiperparámetros de exploración p=1 y q=1. Node2Vec permite representar los nodos en un espacio vectorial que preserva las relaciones de similitud y el contexto entre ellos. Para la optimización de los embeddings, se utilizó el optimizador SparseAdam con una tasa de aprendizaje de 0.01 durante 100 épocas, lo que permitió afinar las representaciones y mejorar la calidad de los embeddings obtenidos. La explicación detallada de cada parámetro se encuentra en la sección de modelos.



A continuación, se generaron embeddings combinados para los nodos, representando tanto a cada tarjeta de crédito como a los comerciantes. Para combinar los embeddings de cada nodo a lo largo de varias iteraciones, se utilizaron dos técnicas: el cálculo de la media aritmética y la concatenación de los vectores resultantes. Esta representación se guardó en el DataFrame original, permitiendo posteriormente dividir el conjunto de datos en entrenamiento, prueba y validación, con una distribución de 70% para entrenamiento, 15% para prueba y 15% para validación.

Como se identificó en la sección de exploración de datos, se detectó un desbalance en la clase objetivo is_fraud. Para abordar esta situación, se optó por utilizar RandomUnderSampler con el fin de reducir el desequilibrio únicamente en el conjunto de datos de entrenamiento. Finalmente se combinaron los embeddings con otras características relevantes como amt, zip, lat, long, city_pop, unix_time, merch_lat, y merch_long con el fin de proporcionar información adicional que ayude al modelo a hacer mejores predicciones.

Modelamiento

Selección métricas de evaluación. Se escogieron accuracy y AUC como métricas de evaluación, debido a su relevancia en problemas de clasificación binaria, como la detección de transacciones fraudulentas donde es necesario identificar correctamente el tipo de transacción mientras se minimizan los errores. Por un lado, el Accuracy mide el porcentaje de predicciones correctas en relación con las transacciones realizadas, proporcionando una visión general del rendimiento del modelo, y el AUC evalúa la capacidad del modelo para distinguir entre clases (fraudulenta y no fraudulenta). Esta métrica es valiosa porque refleja la habilidad del modelo para identificar la transacción sin importar el balance de las clases del problema. La combinación de ambas métricas permite tener un equilibrio entre precisión global y sensibilidad a las transacciones fraudulentas.

Selección de técnicas de modelado. Para alcanzar el objetivo de detección de fraudes, se realizaron cinco iteraciones. La primera utilizó la configuración inicial de embeddings y modelo; en la segunda, se balancearon los datos submuestreando la clase mayoritaria para mejorar la sensibilidad hacia el fraude. En la tercera, se ajustaron los hiperparámetros del modelo Random Forest, optimizando su precisión. En la cuarta iteración, se combinaron el balanceo de clases y la sintonización del modelo con ajustes en los parámetros de los embeddings. Finalmente, en la quita iteración se cambió el modelo, utilizando XGBoost. A continuación, se describe el detalle de cada una de estas iteraciones.

1. **Primera iteración - Configuración inicial**

En esta primera iteración, se configuró el modelo de embeddings Node2Vec sobre un grafo bipartito, en el que los nodos representan dos tipos distintos de entidades: tarjetas de crédito y comerciantes, y las conexiones representan transacciones entre ellos. Con estos embeddings, se entrenó un modelo de clasificación Random Forest sin ajustes adicionales en los hiperparámetros ni en la estructura de los datos. El objetivo de esta iteración fue establecer una línea base del rendimiento del modelo en de AUC y Accuracy, utilizando los embeddings sin modificaciones ni balanceo de clases. Esto permitió observar el rendimiento del sistema con configuraciones estándar y ayudó a identificar posibles áreas de mejora para las siguientes iteraciones.

Para llevar a cabo este procedimiento, se utilizó networkx para construir el grafo bipartito, en donde se representaron las relaciones entre dos entidades: cc_num (tarjetas de crédito) y merchant (comerciantes). Luego, se crearon las conexiones entre estos dos nodos, representando una transacción entre la tarjeta y el comerciante. Estas aristas incluyen el atributo “is_fraud”, que indica si la transacción fue fraudulenta o no. Posteriormente se configuró el modelo Node2vec para la construcción de embeddings de nodos que representen la estructura del grafo, utilizando los siguientes parámetros.

Tabla 1. Configuración del Node2Vec

Configuración Node2Vec	
embedding_dim	128
walk_length	20
context_size	10
walks_per_node	10
num_negative_samples	1
p	1
q	1

Fuente: Elaboración propia

La configuración embedding_dim=128 significa que cada nodo en el grafo será representado por un vector de 128 dimensiones, lo que permite capturar más información sobre las relaciones de cada nodo. La opción walk_length=20 indica que el modelo comenzará un camino aleatorio y recorrerá hasta 20 nodos desde el nodo inicial. El context_size=10 establece que se considerarán 10 nodos cercanos como contexto para cada nodo en cada camino aleatorio. La configuración walks_per_node=10 especifica que se generarán 10 caminos aleatorios distintos desde cada nodo en el grafo durante el proceso de entrenamiento. El num_negative_samples=1 establece que se generará una muestra negativa por cada muestra positiva durante el entrenamiento. Por último, con p = 1 y q = 1 se establece una caminata aleatoria pura, lo que le permite al modelo explorar tanto las conexiones locales como las globales.

Una vez se llevó a cabo la construcción de embeddings con el modelo Node2Vec, se realizó la combinación de los embeddings de la tarjeta y del comerciante mediante la concatenación, es decir, se unieron los dos vectores en uno solo. Posteriormente, se dividieron los datos en conjuntos de entrenamiento, prueba y validación en proporciones del 70%, 15% y 15%, respectivamente, cuyo tamaño se ilustra en la figura 6.

Figura 9. Tamaño del conjunto de entrenamiento, prueba y validación

```
Tamaño del conjunto de entrenamiento: 907672
Tamaño del conjunto de prueba: 194502
Tamaño del conjunto de validación: 194501
```

Fuente: Elaboración propia

A partir de esto, se construyó el modelo Random Forest, basándose en los embeddings combinados de tarjetas y comerciantes, para identificar si una transacción es fraudulenta o no. Este modelo se configuró inicialmente con 100 árboles de decisión. Sin embargo, los resultados del AUC en los conjuntos de prueba y validación indican deficiencias en la capacidad del modelo para detectar transacciones fraudulentas, sugiriendo que se requiere una mejora en el balance del conjunto de datos o en la capacidad del modelo para generalizar, lo cual se abordará en las próximas iteraciones.

Figura 10. Métricas de rendimiento obtenidas en la primera iteración

	Accuracy	AUC
Conjunto de entrenamiento	0.9964	0.6903
Conjunto de prueba	0.9928	0.5455
Conjunto de validación	0.9934	0.5484

Fuente: Elaboración propia

2. **Segunda iteración - Balanceo de clases**

Teniendo en cuenta lo evidenciado en la primera iteración, se procedió a realizar una modificación en el conjunto de datos, balanceando la clase fraudulenta que, como se mencionó en el apartado de preparación de datos, se encontraba desbalanceada. Esta transformación resultó en un total de 10.442 registros en el conjunto de datos. Así las cosas, se mantuvo la misma partición de los datos (70% para entrenamiento, 10% para prueba y 10% para validación) y se utilizó la configuración inicial para la construcción de embeddings y el modelo Random Forest, con lo cual se obtuvieron los siguientes resultados.

Figura 11. Métricas de rendimiento obtenidas en la segunda iteración con balanceo de clases

	Accuracy	AUC
Conjunto de entrenamiento	0.9953	0.9953
Conjunto de prueba	0.7212	0.7034
Conjunto de validación	0.7223	0.6909

Fuente: Elaboración propia

Como resultado del balanceo de clases, se observó una disminución en el accuracy obtenido en los conjuntos de prueba y validación. Sin embargo, se registró una mejora en el AUC. Esta situación

puede atribuirse a que el balanceo de clases puede haber permitido que el modelo se enfoque mejor en la identificación de las transacciones fraudulentas, lo que mejora su capacidad para distinguir entre las clases, reflejándose en un AUC más alto. En las siguientes iteraciones, se realizarán ajustes adicionales en el modelo con el objetivo de seguir optimizando esta métrica.

3. Tercera iteración – Average Embeddings

Para esta iteración, se utilizó el average embedding como parte de la representación de cada transacción en el conjunto de datos, permitiendo capturar características que complementan la información específica de cada transacción. Para esto, el combined_embedding se concatenó con atributos adicionales, tales como el monto de la transacción (amt), código postal (zip), la latitud y la longitud tanto del cliente como del comercio (lat, long, merch_lat, merch_long), tamaño de la población (city_pop) y el tiempo en formato unix (unix_time).

Esta combinación, permite que el modelo tenga una visión más completa de los datos, aprovechando las representaciones capturadas con los embeddings junto con las propiedades específicas de cada transacción. Con estos datos, se entrenó el modelo de Random Forest utilizando la misma configuración de la iteración anterior y se mantuvo la partición de los datos. Los resultados obtenidos reflejan una mejora en la capacidad del modelo para clasificar las transacciones, lo cual se evidencia en las métricas a continuación.

Figura 12. Métricas de rendimiento obtenidas en la tercera iteración

	Accuracy	AUC
Conjunto de entrenamiento	1	1
Conjunto de prueba	0.9449	0.8943
Conjunto de validación	0.9511	0.8781

Fuente: Elaboración propia

El accuracy y el AUC obtenidos muestran una mejora significativa con respecto a las iteraciones anteriores al utilizar características adicionales. Esto sugiere que la inclusión de embeddings le permite al modelo captar relaciones no lineales subyacentes en los datos, mejorando su capacidad de discriminación y reflejándose en un AUC más alto.

4. Cuarta iteración – Sintonización de hiperparámetros

Con los resultados obtenidos en la tercera iteración, se realizó una sintonización de hiperparámetros con el fin de mejorar las métricas obtenidas previamente, dejando la misma preparación utilizada en la iteración anterior. Para esto, se utilizó GridSearch que evalúa múltiples combinaciones de hiperparámetros para seleccionar los que maximicen el desempeño del modelo. En primer lugar, se definió un param_grid para explorar diferentes configuraciones del modelo Random Forest, estas combinaciones incluyeron variaciones en el número de árboles (n_estimators), la profundidad máxima (max_depth), los parámetros de división (min_samples_split) y el número mínimo de muestras en la hoja (min_samples_leaf).

Usando los mejores hiperparámetros encontrados, se entrenó el modelo Random Forest, donde se realizaron predicciones sobre los conjuntos de prueba y validación. En los resultados obtenidos, se puede observar que, aunque el modelo clasifica correctamente la mayoría de las transacciones y tiene un buen rendimiento, las mejores métricas se obtuvieron al realizar el average de los embeddings en la tercera iteración.

Figura 13. Métricas de rendimiento obtenidas en la cuarta iteración

	Accuracy	AUC
Conjunto de prueba	0.9187	0.8881
Conjunto de validación	0.9195	0.8699

Fuente: Elaboración propia

5. Quinta iteración – XGBoost

Para la quinta iteración partimos de los métodos de las iteraciones previas en los cuales se vio una mejora significativa de la clasificación del modelo, esto es, aplicar un modelo de embedding para los nodos de Node2Vec en un grafo bipartito, los parámetros con los cuales se configuró el embedding fueron: Dimensiones: 128; Longitud de los paseos: 20; número de paseos: 80, p=1 y q=1, lo anterior con una ventana de 10 y un batch Word de 4, en un entrenamiento total de 10 épocas. Posterior al entrenamiento de los embeddings de nodos, se procedió a calcular los embeddings de las aristas, que son una combinación entre los embeddings de los nodos que conectan las aristas, en este cálculo se utilizó el Average Embedding que básicamente es el promedio ponderado de los embeddings de los nodos y da como resultado el embedding de la arista.

Como se pudo observar en los puntos anteriores la clase de fraude positivo tenía un desbalanceo significativo con relación a la clase de fraude negativo, de manera que era necesario utilizar una técnica de balanceo, se usó SMOTE para ello, sin embargo previo al balanceo se vio la necesidad de usar un StandardScaler para normalizar los datos ya que un modelo como el XGBoost es sensible a la escala de las características, lo que aporta adicionalmente, una disminución en el sesgo del modelo. Por último, se entrenó el modelo con XGboost, con los parámetros descritos en la tabla a continuación, y el resultado del AUC y Accuracy fue significativamente mejor que en las iteraciones previas:

Hiperparámetros del modelo	**Valor**	**Observaciones**
objective	binary:logistic	Clasificación binaria
max_depth	max_depth: 6	Se fue aumentando a medida de las iteraciones, una profundidad más alta podría sobre ajustar el modelo.
learning_rate	0.01	
n_estimators	1000	Árboles usados
min_child_weight	1	Evitamos que se creen ramas con muy pocos datos.
subsample	0.8	Proporción de filas
colsample_bytree	0.8	Proporción de columnas
gamma	0.1	Controla la regularización L2.
scale_pos_weight	1	Este se dejó en uno porque ya habíamos balanceado las clases

Figura 14. Métricas de rendimiento obtenidas en la quinta iteración

	Accuracy	AUC
Conjunto de prueba	0.9214	0.9784
Conjunto de validación	0.9220	0.9784

Fuente: Elaboración propia

6. Evaluación del modelo.

Figura 12. Métricas de rendimiento obtenidas en el proyecto

	Iteración 1		Iteración 2		Iteración 3		Iteración 4		Iteración 5	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
Conjunto de entrenamiento	0.9964	0.6903	0.9953	0.9953	1	1	1	1	0.99	0.99
Conjunto de prueba	0.9928	0.5455	0.7212	0.7034	0.9449	0.8943	0.9187	0.8881	0.9214	0.9784
Conjunto de validación	0.9934	0.5484	0.7223	0.6909	0.9511	0.8781	0.9195	0.8699	0.9220	0.9784

Fuente: Elaboración propia

A lo largo de las cinco iteraciones, se observaron mejoras notables en las métricas de evaluación de precisión (accuracy) y área bajo la curva ROC (AUC) para los conjuntos de prueba y validación, reflejando el impacto de los ajustes realizados en el modelo. En la primera iteración, el modelo alcanzó un accuracy elevado en el conjunto de entrenamiento (0.9964) pero un AUC bajo en el conjunto de prueba (0.5455), indicando una baja capacidad de discriminación. En la segunda iteración, tras balancear las clases, el AUC aumentó en ambos conjuntos, especialmente en el conjunto de prueba (0.7034), aunque el accuracy disminuyó, lo que demuestra un mejor enfoque en la identificación de fraudes.

La tercera iteración, que incorporó average embeddings y características adicionales, mejoró tanto el accuracy (0.9187 en prueba) como el AUC (0.8881), reflejando una captura de relaciones más complejas entre las transacciones. La cuarta iteración consolidó la estabilidad del modelo, y en la quinta iteración, con el escalado de datos y el uso de XGBoost, el AUC en los conjuntos de prueba y validación alcanzó su punto máximo (0.9784), evidenciando una mejora significativa en la capacidad de clasificación del modelo para distinguir entre transacciones fraudulentas y no fraudulentas.

7. Conclusiones

En este proyecto, se implementó un enfoque basado en grafos y embeddings para la identificación de transacciones fraudulentas, con Node2Vec como técnica central para construir representaciones vectoriales (embeddings) de las conexiones entre nodos. El proceso se desarrolló a través de varias iteraciones, ajustando parámetros para optimizar la precisión del modelo y su capacidad de

generalización. Los resultados obtenidos muestran que el enfoque basado en grafos, combinado con técnicas de aprendizaje automático, es efectivo para detectar fraudes en transacciones.

La transformación que demostró mejorar significativamente los resultados obtenidos en este proyecto fue el balanceo de clases. En particular, el balanceo de las clases permitió que el modelo priorizara la identificación de transacciones fraudulentas, aumentando así su capacidad para detectar correctamente la clase minoritaria (fraude). Aunque se observó una leve disminución en el accuracy en comparación con la primera iteración, se logró un incremento notable en el AUC. Este aumento sugiere que el modelo fue capaz de discriminar mejor entre transacciones fraudulentas y no fraudulentas, aumentando la sensibilidad del modelo a la clase minoritaria (fraude).

Tras el balanceo de los datos, la construcción de average embeddings combinados con características adicionales como el monto y la ubicación geográfica de la transacción resultó en una mejora notable en los valores de accuracy y AUC en los conjuntos de prueba y validación. Esto indica que esta construcción ayudó al modelo a capturar relaciones más complejas en los datos, mejorando su capacidad para distinguir transacciones fraudulentas y no fraudulentas, lo cual se refleja en un AUC más alto en la tercera iteración.

Finalmente, en la quinta iteración se identificó que el escalado de los datos, junto con el uso de un modelo de clasificación más robusto como XGBoost, contribuyó a una mejora significativa en el desempeño general del modelo. Estas mejoras permitieron optimizar la precisión en la clasificación y fortalecer la capacidad del modelo para identificar patrones complejos en las transacciones, logrando una clasificación más efectiva entre transacciones fraudulentas y no fraudulentas.

Informe de participación

Actividad	Quién lo realizó
Selección de datos y planteamiento del problema	Todos
Exploración de los datos	Todos
Iteración 1: Configuración inicial de Embeddings	Daniela Parra
Iteración 2: Balanceo de los datos	Manuel Santiago Hernández
Iteración 3: Modificación de los Embeddings	Lina Varón
Iteración 4: Sintonización Random Forest	Paula Jaimes
Iteración 5: XGBoost	Jenny Ortiz
Conclusiones	Todos

BIBLIOGRAFÍA

Credit Card Transactions Fraud Detection Dataset . Kaggle. Simulated Credit Card Transactions generated using Sparkov. https://www.kaggle.com/datasets/kartik2112/fraud-detection

Zhang, J., & Chen, S. (2019). Community detection in bipartite networks based on Louvain method. In *2019 IEEE International Conference on Communications, Information System and Computer Engineering.* https://journals.aps.org/pre/abstract/10.1103/PhysRevE.90.012805

Informe de transunion 2023 , sobre el fraude omnicanal. Transunion.co https://www.transunion.co/content/dam/transunion/co/business/collateral/report/Reporte_Anual_Tendencias_Fraude_2023_Colombia.pdf