TRABAJO DE GRADO
Opción Seminario-Diplomado.

IMPLEMENTACIÓN DE ALGORITMOS DE MACHINE LEARNING PARA LA DETECCIÓN DE FRAUDES FINANCIEROS INTERNOS EN EJERCICIO DE LA AUDITORÍA TI

Corporación Universitaria Remington.
facultad Ingeniería
Ingeniería de sistemas

Estudiante:
Carlos Andres Ibañez Martinez

Tutor: Juan Carlos Briñez de León

Opción de Trabajo de grado Seminario-Diplomado.
2024.

Dedicatoria

A todos los que han sembrado en mí la semilla del conocimiento y la curiosidad. A mis profesores, por su dedicación y pasión en la enseñanza, y a mis compañeros de Seminario , por su apoyo y colaboración constante. Este trabajo es un reflejo de nuestro esfuerzo colectivo y de la esperanza en un futuro más sostenible. A mi familia, por ser mi mayor inspiración y motivación, siempre alentándome a crecer y aprender.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han hecho posible la realización de este trabajo.

En primer lugar, agradezco a mis profesores y mentores, quienes con su dedicación y sabiduría han guiado mi camino académico. Su apoyo incondicional y sus valiosos consejos han sido fundamentales para el desarrollo de este proyecto.

A mis compañeros del Seminario , gracias por su colaboración, entusiasmo y por compartir ideas que enriquecieron este trabajo. Juntos hemos creado un ambiente de aprendizaje y creatividad que siempre recordaré.

Tabla de Contenidos

Resumen

En este proyecto se desarrolló un modelo para la detección de fraude en transacciones financieras mediante la aplicación de algoritmos de aprendizaje automático, específicamente el algoritmo de clustering K-means. El trabajo comenzó con la preparación y limpieza de un conjunto de datos que contenía información sobre transacciones, con el fin de eliminar las columnas irrelevantes y reducir el ruido. Este proceso incluyó la selección de variables clave, como el tipo de comercio, el monto de la transacción, el tipo de tarjeta, la distancia desde el hogar, y si la transacción ocurrió durante el fin de semana. La limpieza de datos permitió enfocar el análisis en las características más importantes para identificar patrones de comportamiento asociados al fraude.

Una vez que los datos estuvieron listos, se aplicó el algoritmo K-means, que agrupa las transacciones en diferentes clústeres o grupos según sus características compartidas. Este algoritmo se eligió por su capacidad para segmentar grandes volúmenes de datos en categorías homogéneas, lo que es particularmente útil para la detección de anomalías en sistemas financieros. Los resultados del clustering mostraron la formación de grupos de transacciones con distintos niveles de riesgo de fraude. Al analizar estos grupos, se pudo observar que ciertas características, como transacciones en comercios de alto riesgo o montos elevados, estaban más presentes en los clústeres que indicaban un mayor riesgo. Esto permitió identificar patrones que podrían señalar comportamientos fraudulentos, facilitando la detección de anomalías en futuras transacciones.

El análisis realizado destacó el valor de la segmentación de datos en auditoría financiera, ya que permite una evaluación proactiva de riesgos y ayuda a identificar transacciones que merecen mayor atención. Con base en estos resultados, el modelo de K-means puede integrarse en un sistema de auditoría para monitorear las transacciones en tiempo real, clasificando cada una en función de su riesgo potencial y generando alertas cuando se detecten patrones sospechosos. Esto no solo mejora la capacidad de respuesta ante posibles fraudes, sino que también optimiza el proceso de monitoreo al priorizar las transacciones de mayor riesgo.

Este proyecto muestra cómo el aprendizaje automático y los análisis de datos pueden aportar significativamente a la seguridad y la eficiencia en sistemas de auditoría de TI, ofreciendo herramientas innovadoras para la protección de la información financiera. Los hallazgos respaldan el uso de técnicas de clustering para mejorar la detección de fraude, abriendo la puerta a futuras investigaciones que podrían incluir la integración de algoritmos supervisados y la mejora de los modelos predictivos para aumentar la precisión y confiabilidad en la identificación de transacciones fraudulentas.

Palabras clave

transacciones financieras, clustering, K-means, aprendizaje automático, auditoría financiera, análisis de datos, anomalías, riesgo de fraude, sistemas de auditoría, segmentación de datos, patrones de comportamiento, seguridad financiera, auditoría de TI.

1. Marco conceptual y contextual

1.1 Contexto:

1.1.1 Sistemas de recomendación.

Los sistemas de recomendación son herramientas analíticas que utilizan datos históricos para predecir las preferencias de los usuarios y ofrecer sugerencias personalizadas. En el contexto actual, donde las organizaciones están inmersas en procesos de transformación digital, la consolidación de datos se ha vuelto fundamental. Estos sistemas permiten a las empresas analizar decisiones pasadas y las variables asociadas, optimizando así la experiencia del cliente (Ricci, 2015)

La importancia de estos sistemas radica en su capacidad para manejar grandes volúmenes de datos y extraer patrones significativos que pueden influir en las decisiones comerciales. Por ejemplo, en el ámbito del comercio electrónico, los sistemas de recomendación pueden aumentar significativamente las ventas al sugerir productos relevantes a los consumidores (Adomavicius & Tuzhilin, 2005)

. En el caso específico de la detección de fraude, estos sistemas se pueden adaptar para identificar comportamientos anómalos en transacciones financieras, lo que resulta crucial para la seguridad y eficiencia operativa (Zhang et al., 2019)

1.1.2 Algoritmos de Machine learning en sistemas de recomendación.

Los algoritmos de machine learning son fundamentales para mejorar la eficacia de los sistemas de recomendación en la detección de fraudes. Existen diversas técnicas, como el filtrado colaborativo y los modelos basados en contenido, que permiten analizar patrones complejos en los datos transaccionales (Bennett & Lanning, 2007)

En particular, algoritmos como K-means y Random Forest han demostrado ser efectivos para clasificar transacciones y detectar anomalías que podrían indicar fraude (Zhang et al., 2023)

El uso de técnicas como el aprendizaje supervisado y no supervisado permite a las organizaciones identificar comportamientos sospechosos y responder proactivamente a posibles fraudes. Esto es especialmente relevante en el ámbito financiero, donde la rapidez y precisión en la detección son críticas (Adomavicius & Tuzhilin, 2005)

1.2 Descripción de caso de estudio.

El proyecto sobre **detección de fraude financiero en auditoría TI** se centra en desarrollar un modelo que utilice algoritmos de machine learning para identificar transacciones fraudulentas. Este modelo se basa en un conjunto de datos que incluye variables como:

- **Tipo de transacción**: Clasificación entre compras online y físicas.
- **Monto**: Cantidad involucrada en cada transacción.
- **Frecuencia**: Número de transacciones realizadas por un cliente en un periodo específico.

- **Ubicación geográfica**: Datos sobre dónde se realizan las transacciones.
- **Hora del día**: Identificación si las transacciones ocurren durante horas pico o no.

Estas variables son esenciales para entender mejor los patrones asociados con el fraude y facilitar una auditoría más efectiva (García et al., 2020). La necesidad del proyecto radica en la creciente sofisticación del fraude financiero y la importancia de contar con herramientas avanzadas para su detección.

1.3 Pregunta Problema

¿Cómo desarrollar una estrategia computacional para la detección de fraude financiero utilizando algoritmos de Machine Learning dentro del contexto de auditoría TI?

1.4 Hipótesis:

El análisis computacional aplicado a los datos financieros mediante algoritmos como K-means permitirá implementar un sistema eficaz para detectar fraudes dentro del ámbito de auditoría TI. Esto contribuirá a una mayor seguridad y eficiencia operativa.

2. Objetivos:

2.1 Objetivo general.

Implementar una estrategia computacional para la detección de fraude financiero utilizando algoritmos de Machine Learning dentro del ámbito de auditoría TI.

2.2 Objetivos específicos.

- Caracterizar y procesar los datos relevantes para facilitar decisiones informadas.
- Implementar un algoritmo de Machine Learning para la identificación y clasificación de transacciones sospechosas.
- Evaluar y analizar el desempeño del algoritmo implementado para mejorar la toma de decisiones.
- Validar el funcionamiento del sistema utilizando datos nuevos.

3. Desarrollo e implementación del aprendizaje

En esta sección, describo la metodología que implementé y los resultados obtenidos al aplicar técnicas de aprendizaje no supervisado, específicamente el algoritmo de clustering K-means, en el contexto de auditoría de TI para detectar anomalías en transacciones financieras internas. Mi objetivo fue identificar patrones financieros fuera de lo común que pudieran indicar actividades inusuales, potencialmente relacionadas con fraudes, errores o accesos no autorizados.

Esta es una tarea que realizamos cotidianamente quienes trabajamos en el área de auditoría de tecnología de la información, donde el análisis continuo es fundamental para asegurar la integridad y la seguridad de los sistemas. El uso de un algoritmo como K-means permite no solo identificar transacciones financieras sospechosas, sino también analizar otros datos relevantes, como registros de bases de datos (logs), actividades de usuarios y posibles agrupaciones de eventos relacionados con ciberataques o amenazas. Este enfoque amplía la capacidad de monitoreo y facilita una detección temprana de riesgos en múltiples frentes de seguridad dentro de la infraestructura tecnológica.

3.1 Preparación y análisis de los datos

Estructura del Dataset

- **transaction_id**: Identificador único de cada transacción. Es fundamental para rastrear y diferenciar cada operación.
- **customer_id**: Identificador del cliente, que permite asociar múltiples transacciones a un mismo usuario y analizar patrones de comportamiento.
- **card_number**: Número de tarjeta.
- **timestamp**: Fecha y hora de la transacción, esencial para evaluar la temporalidad y frecuencia de las transacciones.
- **merchant_category, merchant_type, merchant:** Información sobre el comercio donde se realizó la transacción. Estos campos ayudan a identificar tipos de transacciones que pueden tener un riesgo diferente según la categoría del comercio
- **amount:** Monto de la transacción, uno de los factores más críticos para detectar irregularidades**.**
- **currency y country:** Moneda y país de la transacción, útiles para analizar si hay transacciones en lugares inusuales para el cliente o en países de alto riesgo.
- **device, channel y device_fingerprint:** Información del dispositivo y canal desde el cual se realiza la transacciónTambién, el device_fingerprint permite verificar si la transacción se realizó desde un dispositivo previamente usado por el cliente.
- **ip_address:** Dirección IP desde la cual se realiza la transacción. Junto con la ubicación, es útil para identificar conexiones sospechosas.

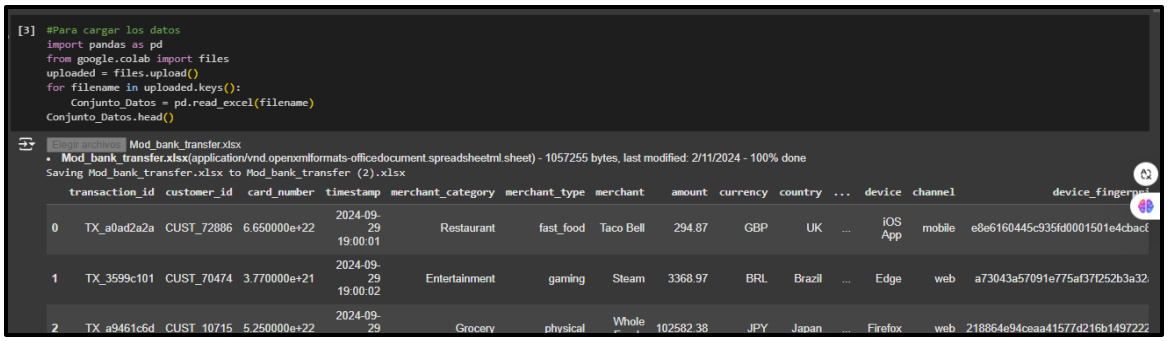
- **distance_from_home:** Distancia desde la ubicación habitual del cliente. Un valor alto podría indicar un comportamiento inusual.
- **high_risk_merchant:** Indicador de si el comercio es considerado de alto riesgo, lo que puede incrementar el índice de sospecha**.**
- **transaction_hour y weekend_transaction**: Hora y si la transacción se realizó en fin de semana, características que ayudan a identificar transacciones en momentos de bajo tráfico o patrones inusuales de tiempo.
- **velocity_last_hour:** Historial de la cantidad y el valor total de transacciones del cliente en la última hora**.**
- **is_fraud:** Etiqueta de la transacción como fraudulenta o no sirve como base para evaluar y validar el modelo.

Se pretende evaluar el dataset para detectar y priorizar las transacciones que representan mayores riesgos financieros, mejorando así la precisión y eficacia del proceso de auditoría de TI. Esto permite que el equipo se concentre en las transacciones de alto riesgo y ayude a prevenir potenciales pérdidas por fraude o errores.

Cargando y visualizando datos en Python (Dataframes)

El fragmento de código presentado inicia un proceso básico en el análisis de datos con Python: la carga y exploración inicial de un conjunto de datos. En primer lugar, se importa la librería Pandas, una herramienta fundamental para la manipulación y análisis

de datos estructurados. A continuación, se carga un archivo de Excel (o una forma utilizando la función de Pandas, transformando los datos del archivo en un DataFrame.



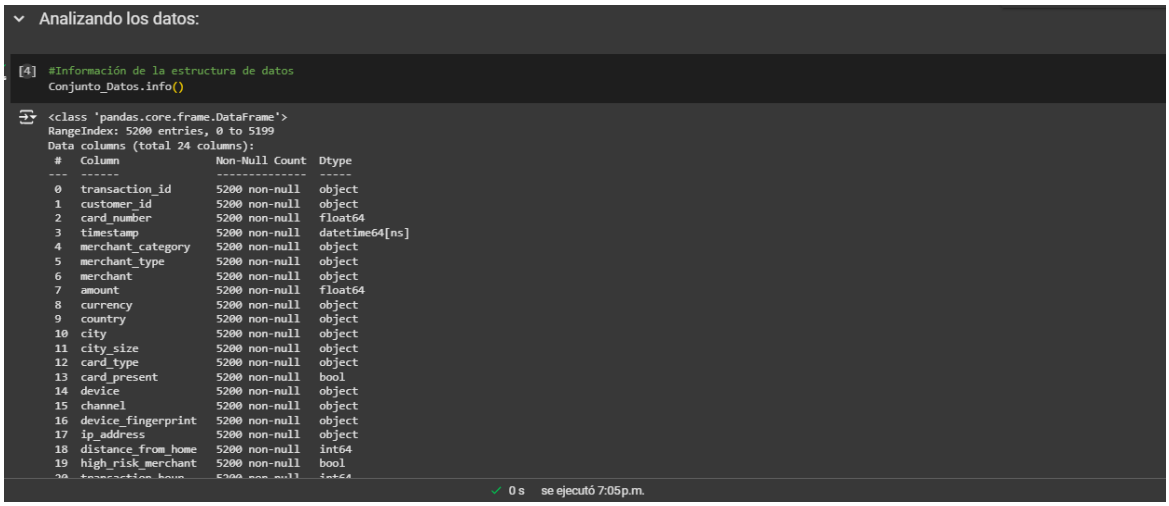
```
[3] #Para cargar los datos
    import pandas as pd
    from google.colab import files
    uploaded = files.upload()
    for filename in uploaded.keys():
        Conjunto_Datos = pd.read_excel(filename)
    Conjunto_Datos.head()
```

Elegir archivos | Mod_bank_transfer.xlsx

- **Mod_bank_transfer.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 1057255 bytes, last modified: 2/11/2024 - 100% done

Saving Mod_bank_transfer.xlsx to Mod_bank_transfer (2).xlsx

	transaction_id	customer_id	card_number	timestamp	merchant_category	merchant_type	merchant	amount	currency	country	...	device	channel	device_fingerprint
0	TX_a0ad2a2a	CUST_72886	6.650000e+22	2024-09-29 19:00:01	Restaurant	fast_food	Taco Bell	294.87	GBP	UK	...	iOS App	mobile	e8e6160445c935fd0001501e4cbac
1	TX_3599c101	CUST_70474	3.770000e+21	2024-09-29 19:00:02	Entertainment	gaming	Steam	3368.97	BRL	Brazil	...	Edge	web	a73043a57091e775af37f252b3a32
2	TX_a9461c6d	CUST_10715	5.250000e+22	2024-09-29	Grocery	physical	Whole	102582.38	JPY	Japan	...	Firefox	web	218864e94ceaa41577d216b1497222



Analizando los datos:

```
[4] #Información de la estructura de datos
    Conjunto_Datos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5200 entries, 0 to 5199
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   transaction_id       5200 non-null   object
 1   customer_id          5200 non-null   object
 2   card_number          5200 non-null   float64
 3   timestamp            5200 non-null   datetime64[ns]
 4   merchant_category    5200 non-null   object
 5   merchant_type        5200 non-null   object
 6   merchant             5200 non-null   object
 7   amount               5200 non-null   float64
 8   currency             5200 non-null   object
 9   country              5200 non-null   object
 10  city                 5200 non-null   object
 11  city_size            5200 non-null   object
 12  card_type            5200 non-null   object
 13  card_present         5200 non-null   bool
 14  device               5200 non-null   object
 15  channel              5200 non-null   object
 16  device_fingerprint   5200 non-null   object
 17  ip_address           5200 non-null   object
 18  distance_from_home   5200 non-null   int64
 19  high_risk_merchant   5200 non-null   bool
```

✓ 0 s se ejecutó 7:05 p.m.

- **Total de Entradas**: El DataFrame tiene 5,200 filas, numeradas de 0 a 5,199.
- **Total, --de Columnas:** Hay 24 columnas en el DataFrame, cada una listada con su nombre y tipo de dato.
- **Tipos de Datos:** Los tipos de datos incluyen object (para datos de texto o mixtos), float64 (para números decimales), datetime (para fechas y horas) y bool (para valores booleanos).

Estadísticas

```
[5] #Análisis de los datos
    Conjunto_Datos.describe()
```

	card_number	timestamp	amount	distance_from_home	transaction_hour
count	5.200000e+03	5200	5.200000e+03	5200.000000	5200.000000
mean	3.805965e+22	2024-09-29 19:30:38.560192256	5.615320e+04	0.375962	0.036346
min	3.710000e+14	2024-09-29 19:00:01	8.000000e-02	0.000000	0.000000
25%	3.790000e+21	2024-09-29 19:15:20.750000128	3.636450e+02	0.000000	0.000000
50%	4.770000e+22	2024-09-29 19:30:27	1.240460e+03	0.000000	0.000000
75%	5.890000e+22	2024-09-29 19:46:09.249999872	2.394238e+04	1.000000	0.000000
max	7.000000e+22	2024-09-29 20:01:14	4.727095e+06	1.000000	1.000000
std	2.519287e+22	NaN	2.049853e+05	0.484417	0.187168

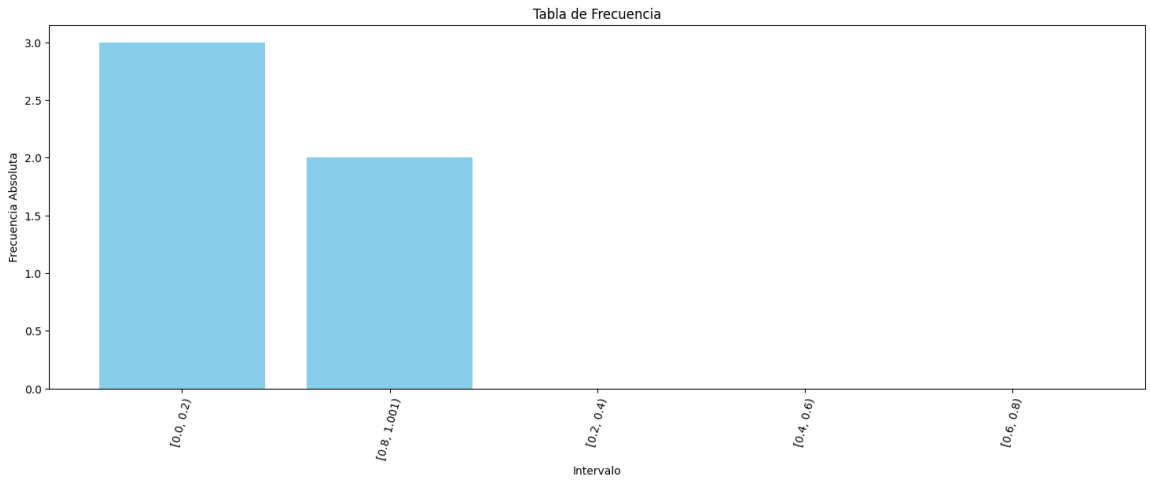
¿Qué podemos inferir de estos resultados?

- Tenemos datos de 5200 transacciones.
- Los montos de las transacciones varían significativamente, desde valores muy bajos hasta valores muy altos.
- La mayoría de las transacciones se realizaron cerca del domicilio del cliente
- El timestamp proporciona información sobre cuándo se realizaron las transacciones.

Realizamos la limpieza del conjunto de datos de transacciones, eliminando las columnas irrelevantes y conservando únicamente la información necesaria para el análisis.

```
#Quitando columnas indeseadas
Conjunto_Datos = Conjunto_Datos.drop(['transaction_id', 'card_number', 'customer_id', 'timestamp',
                                      'merchant_type', 'merchant', 'currency', 'country',
                                      'city_size', 'device', 'device_fingerprint',
                                      'ip_address', 'velocity_last_hour', 'city', 'transaction_hour'], axis=1)


#resumen de los datos
Conjunto_Datos.head(10)
```

	merchant_category	amount	card_type	card_present	channel	distance_from_home	high_risk_merchant	weekend_transaction	is_fraud
0	Restaurant	294.87	Platinum Credit	False	mobile	0	False	False	False
1	Entertainment	3368.97	Platinum Credit	False	web	1	True	False	True
2	Grocery	102582.38	Platinum Credit	False	web	0	False	False	False
3	Gas	630.60	Premium Debit	False	mobile	0	False	False	False
4	Healthcare	724949.27	Basic Debit	False	web	1	False	False	VERDADERO
5	Education	11.76	Platinum Credit	False	web	1	False	False	True
6	Grocery	2606.19	Platinum Credit	False	mobile	0	False	False	False
7	Travel	828.33	Platinum Credit	False	web	0	True	False	False
8	Healthcare	104921.00	Premium Debit	False	web	0	False	False	False

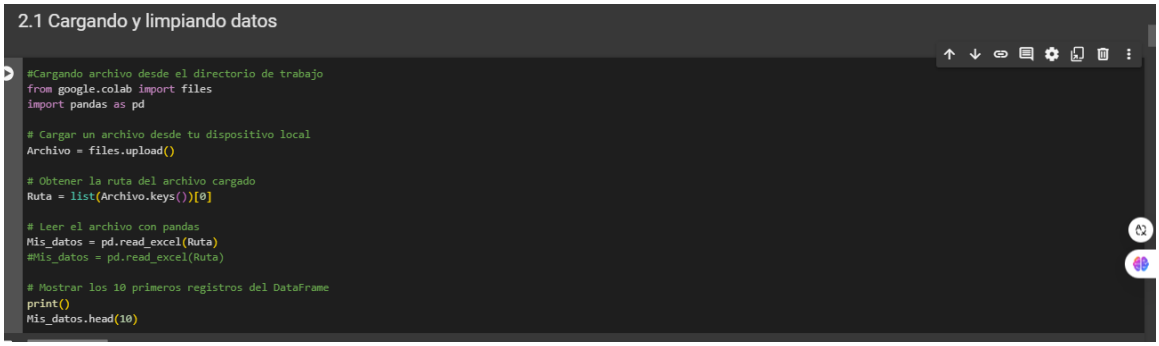


La imagen muestra un histograma de frecuencias absolutas donde la mayoría de los datos se concentran en los intervalos extremos: el rango [0.0, 0.2] tiene la frecuencia más alta (3) y el rango [0.8, 1.0] tiene una frecuencia de 2. No hay datos en los intervalos intermedios, lo que sugiere una polarización en los valores.

3.2 Modelo de toma de decisiones

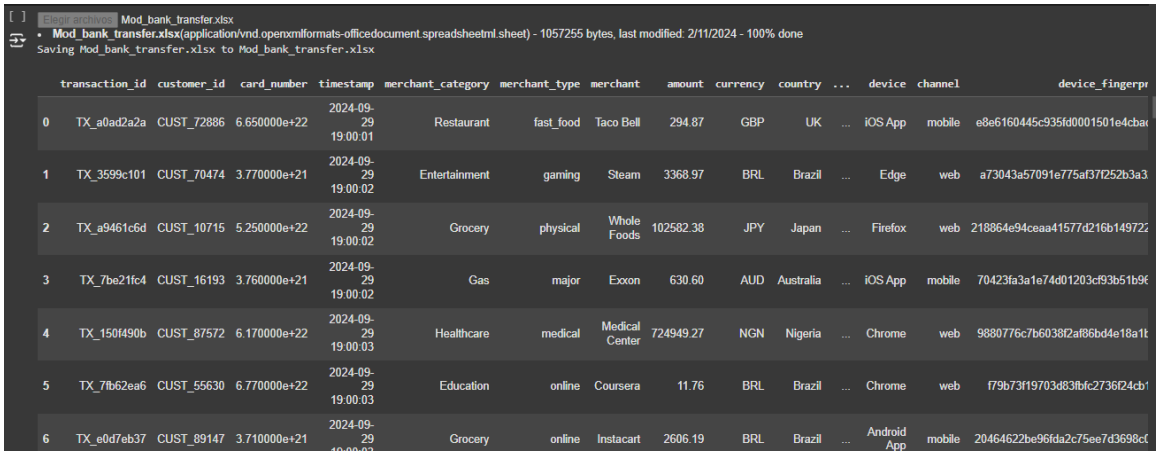
Implementación de Algoritmo Clustering.

Cargamos los datos del dataset y, para ello, es importante importar la librería Pandas, que proporciona funciones esenciales para manipular y analizar datos de manera eficiente.



2.1 Cargando y limpiando datos

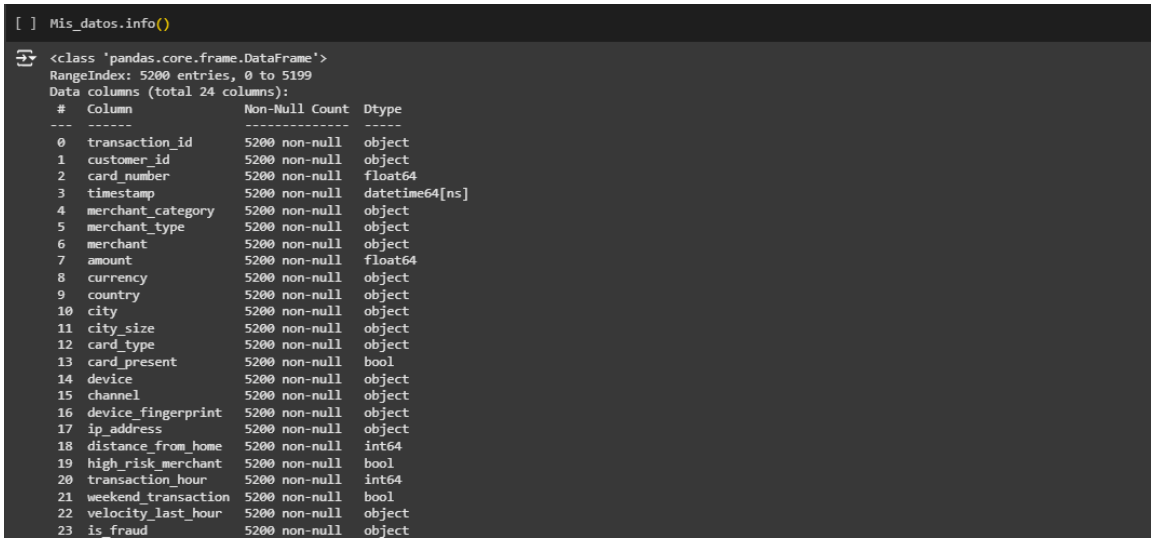
```
#Cargando archivo desde el directorio de trabajo
from google.colab import files
import pandas as pd

# Cargar un archivo desde tu dispositivo local
Archivo = files.upload()

# Obtener la ruta del archivo cargado
Ruta = list(Archivo.keys())[0]

# Leer el archivo con pandas
Mis_datos = pd.read_excel(Ruta)
#Mis_datos = pd.read_excel(Ruta)

# Mostrar los 10 primeros registros del DataFrame
print()
Mis_datos.head(10)
```



```
[ ]
Elegir archivos  Mod_bank_transfer.xlsx
• Mod_bank_transfer.xlsx(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 1057255 bytes, last modified: 2/11/2024 - 100% done
Saving Mod_bank_transfer.xlsx to Mod_bank_transfer.xlsx
```

	transaction_id	customer_id	card_number	timestamp	merchant_category	merchant_type	merchant	amount	currency	country	...	device	channel	device_fingerpr
0	TX_a0ad2a2a	CUST_72886	6.650000e+22	2024-09-29 19:00:01	Restaurant	fast_food	Taco Bell	294.87	GBP	UK	...	iOS App	mobile	e8e6160445c935fd0001501e4cba
1	TX_3599c101	CUST_70474	3.770000e+21	2024-09-29 19:00:02	Entertainment	gaming	Steam	3368.97	BRL	Brazil	...	Edge	web	a73043a57091e775af37f252b3a3
2	TX_a9461c6d	CUST_10715	5.250000e+22	2024-09-29 19:00:02	Grocery	physical	Whole Foods	102582.38	JPY	Japan	...	Firefox	web	218864e94ceaa41577d216b149722
3	TX_7be21fc4	CUST_16193	3.760000e+21	2024-09-29 19:00:02	Gas	major	Exxon	630.60	AUD	Australia	...	iOS App	mobile	70423fa3a1e74d01203cf93b51b9
4	TX_150f490b	CUST_87572	6.170000e+22	2024-09-29 19:00:03	Healthcare	medical	Medical Center	724949.27	NGN	Nigeria	...	Chrome	web	9880776c7b6038f2af86bd4e18a1
5	TX_7fb62ea6	CUST_55630	6.770000e+22	2024-09-29 19:00:03	Education	online	Coursera	11.76	BRL	Brazil	...	Chrome	web	f79b73f19703d83fbfc2736f24cb
6	TX_e0d7eb37	CUST_89147	3.710000e+21	2024-09-29	Grocery	online	Instacart	2606.19	BRL	Brazil	...	Android App	mobile	20464622be96fda2c75ee7d3698c

Se realiza una descripción de los datos de la tabla para identificar los tipos de datos presentes y, de este modo, determinar las acciones de análisis y procesamiento adecuadas para cada columna.



```
[ ] Mis_datos.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5200 entries, 0 to 5199
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   transaction_id       5200 non-null   object
 1   customer_id          5200 non-null   object
 2   card_number          5200 non-null   float64
 3   timestamp            5200 non-null   datetime64[ns]
 4   merchant_category    5200 non-null   object
 5   merchant_type        5200 non-null   object
 6   merchant             5200 non-null   object
 7   amount               5200 non-null   float64
 8   currency             5200 non-null   object
 9   country              5200 non-null   object
 10  city                 5200 non-null   object
 11  city_size            5200 non-null   object
 12  card_type            5200 non-null   object
 13  card_present         5200 non-null   bool
 14  device               5200 non-null   object
 15  channel              5200 non-null   object
 16  device_fingerprint   5200 non-null   object
 17  ip_address           5200 non-null   object
 18  distance_from_home   5200 non-null   int64
 19  high_risk_merchant   5200 non-null   bool
 20  transaction_hour     5200 non-null   int64
 21  weekend_transaction  5200 non-null   bool
 22  velocity_last_hour   5200 non-null   object
 23  is_fraud             5200 non-null   object
```

En la imagen, el dataset tiene 24 columnas con los siguientes tipos de datos:

- **object**: Se utiliza para columnas de texto o categóricas, como transaction_id, customer_id, merchant, y is_fraud.
- **float64**: Representa números decimales, por ejemplo, card_number y amount.
- **datetime64[ns**: Usado para datos de tiempo en la columna timestamp.
- **int64:** Columnas de números enteros, como distance_from_home y weekend_transaction.

Para continuar, es necesario eliminar las columnas que no aportan valor al análisis y transformar los datos de tipo texto a formatos adecuados para su procesamiento.

```
#Quitando columnas indeseadas
Mis_datos = Mis_datos.drop(['transaction_id', 'card_number', 'customer_id', 'timestamp',
                            'merchant_type', 'merchant', 'currency', 'country',
                            'city_size', 'device', 'device_fingerprint',
                            'ip_address', 'velocity_last_hour', 'city', 'transaction_hour'], axis=1)


#resumen de los datos
Mis_datos.head(10)
```

Como resultado de la eliminación de columnas innecesarias, obtenemos un dataset más limpio y enfocado para el análisis.

	merchant_category	amount	card_type	card_present	channel	distance_from_home	high_risk_merchant	weekend_transaction	is_fraud
0	Restaurant	294.87	Platinum Credit	False	mobile	0	False	False	False
1	Entertainment	3368.97	Platinum Credit	False	web	1	True	False	True
2	Grocery	102582.38	Platinum Credit	False	web	0	False	False	False
3	Gas	630.60	Premium Debit	False	mobile	0	False	False	False
4	Healthcare	724949.27	Basic Debit	False	web	1	False	False	VERDADERO
5	Education	11.76	Platinum Credit	False	web	1	False	False	True
6	Grocery	2606.19	Platinum Credit	False	mobile	0	False	False	False
7	Travel	828.33	Platinum Credit	False	web	0	True	False	False
8	Healthcare	104921.00	Premium Debit	False	web	0	False	False	False
9	Retail	51521.84	Premium Debit	True	pos	1	False	False	True

Para implementar el algoritmo de clustering, es necesario transformar la tabla en una matriz de números enteros que permita su procesamiento. Para ello, convertiremos los valores categóricos en representaciones numéricas.

```
[ ]  # Convertir la columna 'is_fraud' de booleano a texto
     Mis_datos['is_fraud'] = Mis_datos['is_fraud'].map({False: 'NO FRAUDE', True: 'FRUEDE'})
     Mis_datos.head()
```

```
#Convertimos las Otras Variables a Texto

# Convertir la columna 'high_risk_merchant' de booleano a texto
Mis_datos['high_risk_merchant'] = Mis_datos['high_risk_merchant'].map({False: 'NO RIESGO', True: 'RIESGO'})

# Convertir la columna 'weekend_transaction' de booleano a texto
Mis_datos['weekend_transaction'] = Mis_datos['weekend_transaction'].map({False: 'NO', True: 'SI'})

# Convertir la columna 'card_present' de booleano a texto
Mis_datos['card_present'] = Mis_datos['card_present'].map({False: 'NO', True: 'SI'})

Mis_datos.head()
```

Convertimos las variables categóricas a valores enteros

```
[ ]  # Cambiando valores en la variable card_type
     Opciones_card = {
         'Platinum Credit': 1,
         'Premium Debit': 2,
         'Basic Debit': 3,
         'Gold Credit': 4,
         'Basic Credit': 5
     }

     Mis_datos['card_type'] = Mis_datos['card_type'].map(Opciones_card)
```

```
     # Cambiando valores en la variable channel
     Opciones_channel = {
         'mobile': 1,
         'web': 2,
         'pos': 3
     }

     Mis_datos['channel'] = Mis_datos['channel'].map(Opciones_channel)
```

```
[ ]  # Cambiando valores en la variable card_present
     Opciones = {'NO':1,'SI':2}
     Mis_datos['card_present'] = Mis_datos['card_present'].map(Opciones)
     Mis_datos.head()
```

La tabla está lista para ser convertida en una matriz y proceder con el análisis

	merchant_category	amount	card_type	card_present	channel	distance_from_home	high_risk_merchant	weekend_transaction	is_fraud
0	1	294	1	1	1	0	1	1	1
1	2	3368	1	1	2	1	2	1	2
2	3	102582	1	1	2	0	1	1	1
3	4	630	2	1	1	0	1	1	1
5	6	11	1	1	2	1	1	1	2

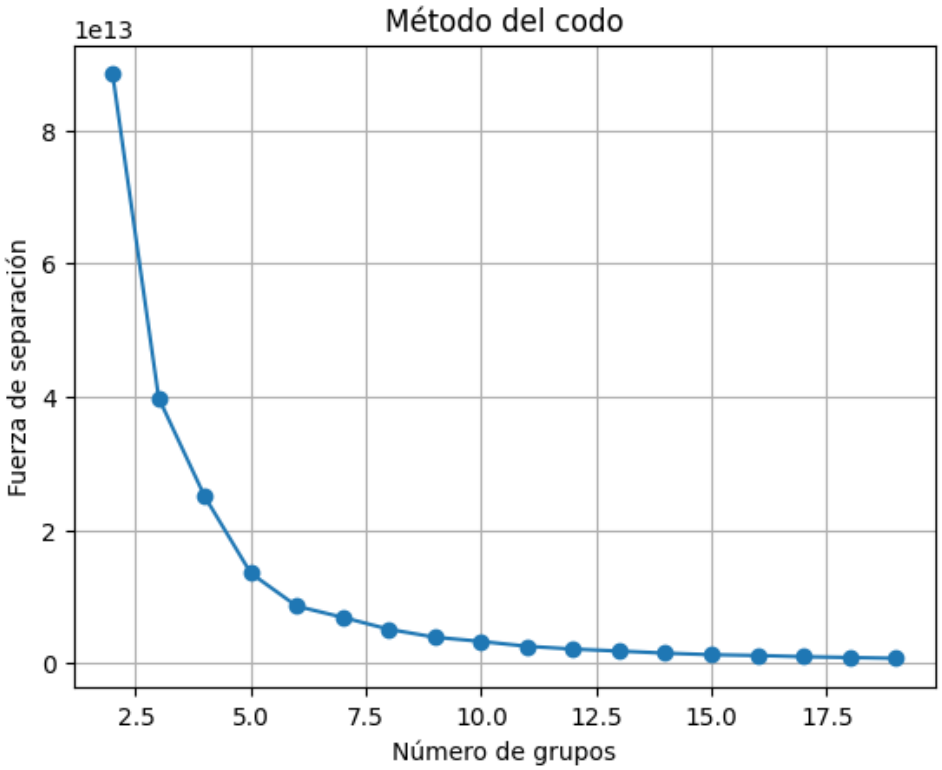
Convertimos la tabla en una Matriz

```
[17] #Convierto los datos a matriz numérica (Matemàtica)
     import numpy as np
     Datos_Array = np.array(Mis_datos)
```

Creando modelo de ML para la segmentación de las transacciones financieras.

```
#Convierto los datos a hoja de cálculo
import numpy as np
import time
Datos_Array = np.array(Mis_datos)

# Creando modelo(KMeans es una librería para hacer clustering)
from sklearn.cluster import KMeans
from sklearn import metrics

# Ejecutar KMeans con diferentes números de clusters
inertia_values = []
silhouette_scores = []
Inicial = 2
Final = 20
k_range = range(Inicial, Final)

for k in k_range:
    kmeans = KMeans(n_clusters=k,n_init = 'auto', random_state=42)
    kmeans.fit(Datos_Array)
    inertia_values.append(kmeans.inertia_)
    silhouette_scores.append(metrics.silhouette_score(Datos_Array, kmeans.labels_))

# Graficar el método del codo
plt.plot(k_range, inertia_values, marker='o')
plt.xlabel('Número de grupos')
plt.ylabel('Fuerza de separación')
plt.title('Método del codo')
plt.grid(True)
plt.show()
```

Con base en la siguiente gráfica y aplicando el método del codo, se deben seleccionar los datos alrededor del punto donde se observa un cambio significativo en la pendiente ("codo"). En este caso, se procederá a dividir los datos en 5 grupos, a que este valor se encuentra cercano al punto de inflexión en la gráfica**.**

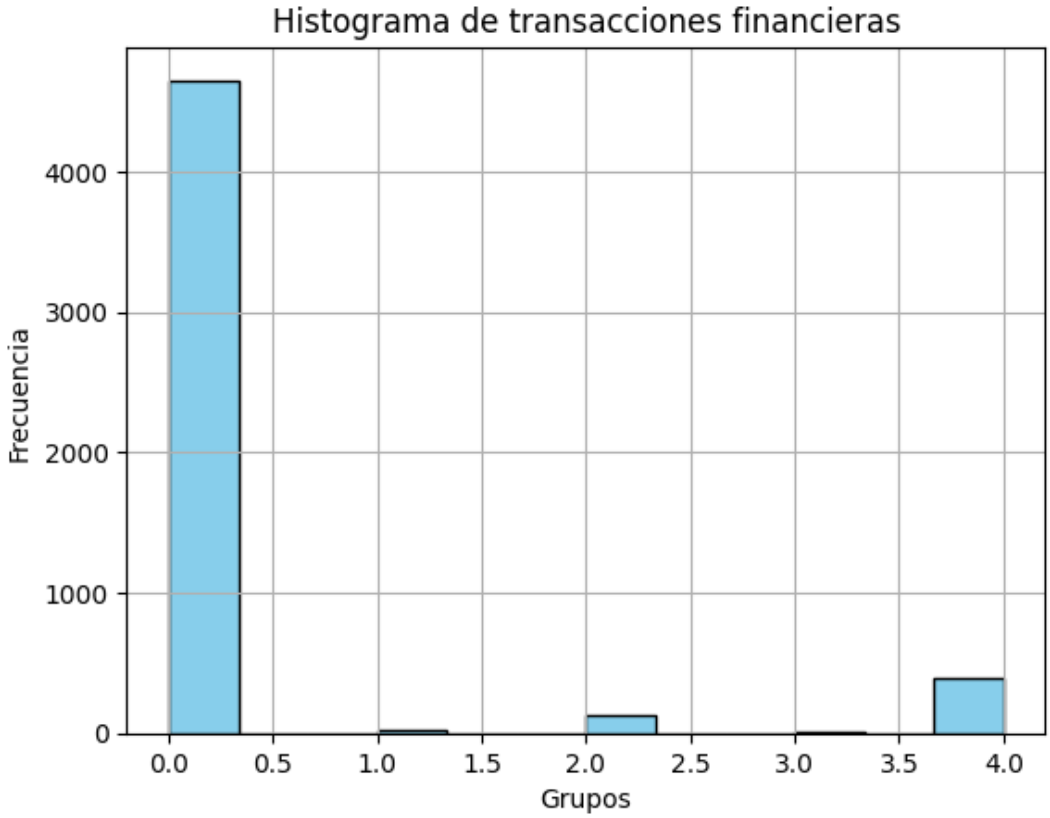


Posterior a esto hemos entrenado nuestra inteligencia artificial la cual aprenderá de los datos entregados; quedando de la siguiente manera

Al estar entrenado nuestro modelo de datos, se nos generará una tabla con los grupos entregados por parte de la inteligencia artificial, quedando de la siguiente manera.

	merchant_category	amount	card_type	card_present	channel	distance_from_home	high_risk_merchant	weekend_transaction	is_fraud	Grupo
0	1	294	1	1	1	0	1	1	1	0
1	2	3368	1	1	2	1	2	1	2	0
2	3	102582	1	1	2	0	1	1	1	0
3	4	630	2	1	1	0	1	1	1	0
5	6	11	1	1	2	1	1	1	2	0
6	3	2606	1	1	1	0	1	1	1	0
7	7	828	1	1	2	0	2	1	1	0
8	5	104921	2	1	2	0	1	1	1	0
9	8	51521	2	2	3	1	1	1	2	0
10	4	34644	2	1	2	0	1	1	1	0

Este grafico nos muestra a más detalle como quedaron repartidos los grupos en nuestros datos

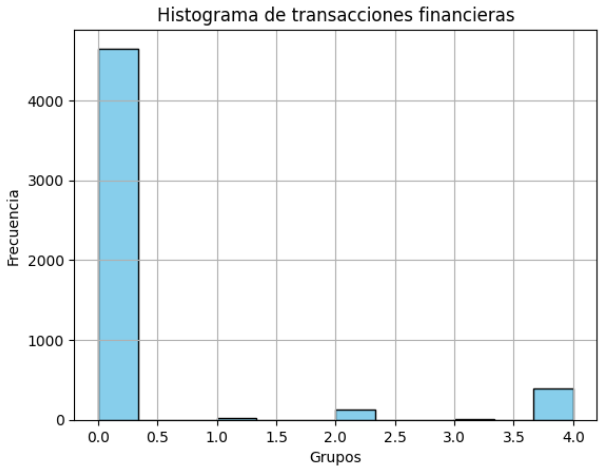


Tenemos la siguiente tabla que nos dará más detalle de cómo están repartidos los grupos y de otras características adicionales.

Centroides:

	merchant_category	amount	card_type	card_present	channel	distance_from_home	high_risk_merchant	weekend_transaction	is_fraud
0	4.391482	1.099259e+04	2.981286	1.110992	1.824048	0.356636	1.256829	1.0	1.249301
1	7.421053	1.688941e+06	2.684211	1.368421	2.210526	0.894737	1.578947	1.0	1.947368
2	5.059701	7.228153e+05	2.514925	1.380597	2.164179	0.701493	1.253731	1.0	1.791045
3	7.000000	3.545082e+06	2.666667	1.500000	2.000000	0.833333	2.000000	1.0	2.000000
4	4.851662	2.300433e+05	2.815857	1.156010	1.869565	0.460358	1.250639	1.0	1.393862

3.3 Análisis de desempeño



- El histograma muestra que la mayoría de las transacciones se encuentran en un único grupo 0, mientras que los demás Grupo tienen una frecuencia significativamente menor.

- Esta concentración en un solo grupos podría indicar que la mayoría de las transacciones siguen un patrón común, mientras que las transacciones en otros grupos podrían estar asociadas con comportamientos anómalos o menos frecuentes.

Análisis de los grupos.

Centroides:

	merchant_category	amount	card_type	card_present	channel	distance_from_home	high_risk_merchant	weekend_transaction	is_fraud
0	4.391482	1.099259e+04	2.981286	1.110992	1.824048	0.356636	1.256829	1.0	1.249301
1	7.421053	1.688941e+06	2.684211	1.368421	2.210526	0.894737	1.578947	1.0	1.947368
2	5.059701	7.228153e+05	2.514925	1.380597	2.164179	0.701493	1.253731	1.0	1.791045
3	7.000000	3.545082e+06	2.666667	1.500000	2.000000	0.833333	2.000000	1.0	2.000000
4	4.851662	2.300433e+05	2.815857	1.156010	1.869565	0.460358	1.250639	1.0	1.393862

Grupo 0

merchant_category: 4.39 (Probablemente representa una categoría de comercio específica).

amount: 10,992.59 Monto promedio bajo en comparación con otros Grupos**.**

high_risk_merchant: 1.25 Podría representar un riesgo moderado de fraude.

is_fraud: 1.25 Indica una tendencia hacia fraude, pero menos que otros Grupo.

Este grupo parece representar transacciones comunes con montos relativamente bajos. Aunque hay cierto riesgo de fraude, no es tan alto como en otros grupos**.**

Grupo 1

amount: 1,688,941 Monto muy alto.

distance_from_home: 0.89 Cercano al hogar del usuario.

is_fraud: 1.95 (Alta probabilidad de fraude.

Este grupo parece representar transacciones de monto alto y una tendencia significativa hacia el fraude, aunque el riesgo asociado al comercio es moderado. Este podría ser un grupo de alto interés para detección de fraude**.**

Grupo 2

amount: 722,815.3 moderadamente alto.

distance_from_home: 0.70 Relativamente cerca del hogar.

is_fraud: 1.79 También una alta probabilidad de fraude.

Este grupo también muestra un riesgo elevado de fraude con montos significativos, aunque menores que en el grupo 1.

Grupo 3

amount: 3,545,082 Monto muy alto, el más alto entre todos.

is_fraud: 2.0 Riesgo máximo de fraude.

Este es el grupo de mayor riesgo y representa las transacciones más caras, todas de alto riesgo. Es muy probable que este grupo esté asociado con actividades fraudulentas.

Grupo 4

amount: 230,043.3 Monto bajo-moderado**.**

is_fraud: 1.39 Moderada tendencia hacia el fraude.

Este grupo representa transacciones con un riesgo moderado y montos intermedios, por lo que podría tener tanto transacciones fraudulentas como legítimas.

Conclusión.

- Grupo 1, 2, muestran una tendencia alta hacia el fraude y deberían ser analizados más a fondo.
- Grupo 3, en particular, presenta transacciones de montos excepcionalmente altos y riesgo máximo de fraude.
- Grupo 0 probablemente contiene la mayoría de las transacciones legítimas, con montos bajos y riesgo moderado.

- Grupo 4 tiene montos intermedios y podría incluir tanto transacciones legítimas como algunas sospechosas.

3.4 Validación del modelo

Con los datos proporcionados, podemos implementar un algoritmo que aproveche el aprendizaje previo para predecir posibles patrones en futuras transacciones.

```python
import numpy as np

# Crear una nueva transacción con los valores inicializados en cero
transaction_New = np.zeros((1, 9))  # Cambiamos a 9 columnas para las variables del modelo

# Pedir al usuario que ingrese los datos de la nueva transacción
transaction_New[0, 8] = float(input('Ingrese si es fraude (1 para Sí, 2 para No): '))
transaction_New[0, 0] = float(input('Ingrese categoría de comercio (1: Restaurant, 2: Entertainment, 3: Grocery, 4: Gas, 5: Healthcare, 6: Education, 7: Travel, 8: Retail): '))
transaction_New[0, 1] = float(input('Ingrese monto de la transacción: '))
transaction_New[0, 2] = float(input('Ingrese tipo de tarjeta (1: Basic Debit, 2: Premium Debit, 3: Platinum Credit, 4: Gold Credit): '))
transaction_New[0, 3] = float(input('¿La tarjeta estuvo presente? (1 para Sí, 2 para No): '))
transaction_New[0, 4] = float(input('Ingrese canal (1: Web, 2: Mobile, 3: POS): '))
transaction_New[0, 5] = float(input('Ingrese distancia desde el hogar (0 para cerca, 1 para lejos): '))
transaction_New[0, 6] = float(input('¿Es un comercio de alto riesgo? (1 para Sí, 2 para No): '))
transaction_New[0, 7] = float(input('¿Es una transacción de fin de semana? (1 para Sí, 2 para No): '))

# Asumimos que 'Modelo_Cluster' es el modelo K-means previamente entrenado
Etiqueta_Cliente = Modelo_Cluster.predict(transaction_New)

# Mostrar el grupo asignado
print('Según los datos de la transacción, el grupo asignado es: ', Etiqueta_Cliente[0])

# Detectar posibles fraudes basándose en el grupo
if Etiqueta_Cliente[0] in [1, 3]:  # Ejemplo: supongamos que grupos 1 y 3 están asociados con un alto riesgo de fraude
    print("Alerta: La transacción pertenece a un grupo de alto riesgo de fraude.")
else:
    print("La transacción pertenece a un grupo de bajo riesgo de fraude.")
```

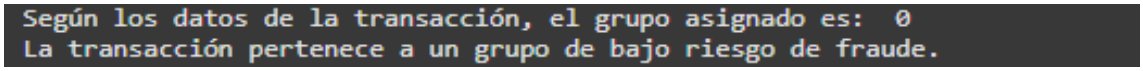
El algoritmo solicitara las entradas de datos al usuario para caracterizar una transacción financiera con el fin de predecir su riesgo de fraude utilizando un modelo de clustering.

1. **Ingresar si es fraude**: El usuario selecciona si la transacción es fraudulenta (1 para "Sí", 2 para "No").
2. **Categoría de comercio**: Se elige el tipo de comercio entre varias opciones
3. **Monto de la transacción**: El usuario ingresa el monto de la transacción (en este caso, 60000).
4. **Tipo de tarjeta**: El usuario selecciona el tipo de
5. **Presencia de la tarjeta**: Se indica si la tarjeta estuvo presente (1 para "Sí", 2 para "No").

6. **Canal**: El usuario selecciona el canal de la transacción (Web, Mobile, o POS).
7. **Distancia desde el hogar**: Indica si la transacción fue realizada cerca (0) o lejos (1) del hogar.
8. **Comercio de alto riesgo**: Se especifica si el comercio es de alto riesgo.
9. **Transacción de fin de semana**: Se indica si la transacción ocurrió en fin de semana.

```
Ingrese si es fraude (1 para Sí, 2 para No): 1
Ingrese categoría de comercio (1: Restaurant, 2: Entertainment, 3: Grocery, 4: Gas, 5: Healthcare, 6: Education, 7: Travel, 8: Retail): 2
Ingrese monto de la transacción: 60000
Ingrese tipo de tarjeta (1: Basic Debit, 2: Premium Debit, 3: Platinum Credit, 4: Gold Credit): 1
¿La tarjeta estuvo presente? (1 para Sí, 2 para No): 1
Ingrese canal (1: Web, 2: Mobile, 3: POS): 1
Ingrese distancia desde el hogar (0 para cerca, 1 para lejos): 0
¿Es un comercio de alto riesgo? (1 para Sí, 2 para No): 1
¿Es una transacción de fin de semana? (1 para Sí, 2 para No): 1
Según los datos de la transacción, el grupo asignado es:  0
La transacción pertenece a un grupo de bajo riesgo de fraude.
```

Después de ingresar estos datos, el sistema asigna a la transacción un grupo de riesgo. En este caso, la transacción se clasifica en el grupo 0 y el sistema indica que "la transacción pertenece a un grupo de bajo riesgo de fraude".



Este flujo de datos y predicción permite evaluar la probabilidad de que una transacción sea fraudulenta en función de las características ingresadas.

4. Conclusiones y trabajos futuros

Conclusiones para la Auditoría TI a los Riesgos Financieros.

1. **Transacciones Concentradas en un grupo Común**: La mayoría de las transacciones en grupo 0 son probablemente legítimas. Este grupo representa el comportamiento transaccional estándar, y su análisis puede servir de referencia para identificar patrones normales.
2. **Grupos Anómalos 1, 2, y 3**: Estos grupos contienen características inusuales, como montos altos, comercio de alto riesgo y alta probabilidad de fraude. Estos grupos son de alto interés en la auditoría, ya que concentran las transacciones que podrían ser fraudulentas.
3. **Riesgo en Canales y Ubicación**: El análisis revela que el canal y la distancia desde el hogar del usuario pueden influir en el riesgo de fraude. Esto indica posibles brechas en la seguridad del sistema de información, especialmente en transacciones remotas o en canales específicos.

Referencias bibliográficas

1. Zhang, Y., et al. (2023). Avances en el uso de inteligencia artificial para la mejora del control y la detección de fraudes en organizaciones. *Semantics Scholar*. https://www.semanticscholar.org/paper/72fcc7447cfb876e61c068da5ea0797123ebe2a7

2. García, R., et al. (2020). Implementación de una herramienta tecnológica que permita realizar el monitoreo centralizado que apoye a la prevención del fraude para transacciones de instituciones financieras. *Semantics Scholar*. https://www.semanticscholar.org/paper/6f461154fb853be9d258c216c8b57e651a2f61d9

3. Bennett, P.N., & Lanning, S.J. (2007). The Netflix Prize. *Proceedings of KDD Cup*. https://www.semanticscholar.org/paper/66946883c9253cb8078ba80396f2faf82b867568

4. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and future research directions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734-749. https://www.semanticscholar.org/paper/ebc9c78eb4bf5f496e34f1e0e76316d2e8bad774

5. Zhang, Y., et al. (2023). Aplicación de Modelos de Aprendizaje Automático en la Detección de Fraudes en Transacciones Financieras. *Semantics Scholar*. https://www.semanticscholar.org/paper/9437aa0246d40dffae20ca24eab3117118eba73f