CAN YOU DETECT FRAUD FROM CUSTOMER TRANSACTIONS?

INFORME FINAL

PRESENTADO POR:

EVELYN ZHARICK SAEZ

GALLEGO

INGENIERÍA AMBIENTAL

PRESENTADO A:

RAÚL RAMOS POLLAN

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERIA

2023

INTRODUCCIÓN

Este informe ejecutivo es la entrega final del curso llamado Inteligencia Artificial para las Ciencias y la Ingeniería donde se expone los resultados finales de la competición que se escogió en Kaggle. Para esto, se realizó una exploración descriptiva del dataset posteriormente se procesaron los datos para dar cuenta de los resultados y por último se expusieron los retos, dificultades y ventajas que se tuvieron en el desarrollo del presente trabajo con las respectivas conclusiones.

PREPROCESAMIENTO DEL DATASET

La mayor parte del tiempo se dedicó a familiarizarse con la forma de obtener los datos de Kaggle. Inicialmente, se intentó cargar los datos desde Google Drive, pero esta opción resultó ineficiente. Se intentó desde un hosting alternativo haciendo el llamado desde colab, pero tampoco funcionó, luego de leer toda la documentación disponible, pudimos crear la API KEY y traer los datos directamente desde la competencia de Kaggle y que funcionara correctamente.

Hemos decidido trabajar con el dataset de la empresa VESTA en la cual se trata de detectar cuando una transacción es fraudulenta. La variable crítica es IsFraud que finalmente se convierte en un booleano que toma valores True o False

El Dataset está compuesto por 4 archivos .CSV (tablas) así:

sample_submission

test_identity

test_transaction

train_identity

train_transaction

De estos, teníamos que elegir con cual trabajaríamos, sin embargo, desde la misma competencia de Kaggle nos indicaban que ambas tablas estaban relacionadas por la clave primaria del código de la transacción, por lo que sabemos desde ya que para el desarrollo final de este trabajo debemos incluir a ambas: **Identity** and **Transactions**.

ANALISIS DE LOS DATOS

TABLA IDENTITY

Las variables en esta tabla son información de identidad:

información de conexión de red (IP, ISP, Proxy, etc.) y firma digital (UA/ navegador/OS/versión, etc.) asociada con las transacciones.

Son recopilados por el sistema de protección contra fraudes de Vesta y los socios de seguridad digital.

(Los nombres de los campos están enmascarados y no se proporcionará el diccionario por pares para la protección de la privacidad y el acuerdo del contrato)

- TransactionID
- id_12 - id_38
- DeviceType
- DeviceInfo

TABLA TRANSACTIONS:

- TransactionDT: timedelta de una fecha y hora de referencia determinada (no una marca de tiempo real). timedelta de una fecha y hora de referencia dada (no una marca de tiempo real). El primer valor de TransactionDT es 86400, que corresponde a la cantidad de segundos en un día (60 * 60 * 24 = 86400), así que creo que la unidad es segundos. Usando esto, sabemos que los datos abarcan 6 meses, ya que el valor máximo es 15811131, que correspondería al día 183"
- TransactionAMT: monto del pago de la transacción en USD
- ProductCD: código de producto, el producto para cada transacción
- card1 - card6: información de la tarjeta de pago, como tipo de tarjeta, categoría de tarjeta, banco emisor, país, etc.
- dirección: dirección addr1 como región de facturación, addr2 como país de facturación
- distancia: distancias entre (no limitadas) la dirección de facturación, la dirección postal, el código postal, la dirección IP, el área telefónica, etc
- P_ y (R__) emaildomain: dominio de correo electrónico del comprador y del destinatario
- C1-C14: conteo, como cuántas direcciones se encuentran asociadas con la tarjeta de pago, etc. El significado real está enmascarado.
- D1-D15: timedelta, como días entre transacciones anteriores, etc.
- M1-M9: coincidencia, como nombres en la tarjeta y dirección, etc.
- Vxxx: características completas diseñadas por Vesta, que incluyen clasificación, conteo y otras relaciones de entidad.

Características categóricas:

ProductCD

card1 - card6

addr1, addr2

P_emaildomain

R_emaildomain

M1 - M9

La tabla más grande corresponde a la de transacciones y es la que tiene información más relevante, por ejemplo, el monto de la transacción la cual podríamos usar para saber el monto total de transacciones que son fraudulentas, cruzándola con la tabla identidad, podríamos conocer desde

que navegador se realizan, o cual franquicia es la más vulnerada (Amex, Visa, Mastercard, etc) por monto o por cantidad de repeticiones.

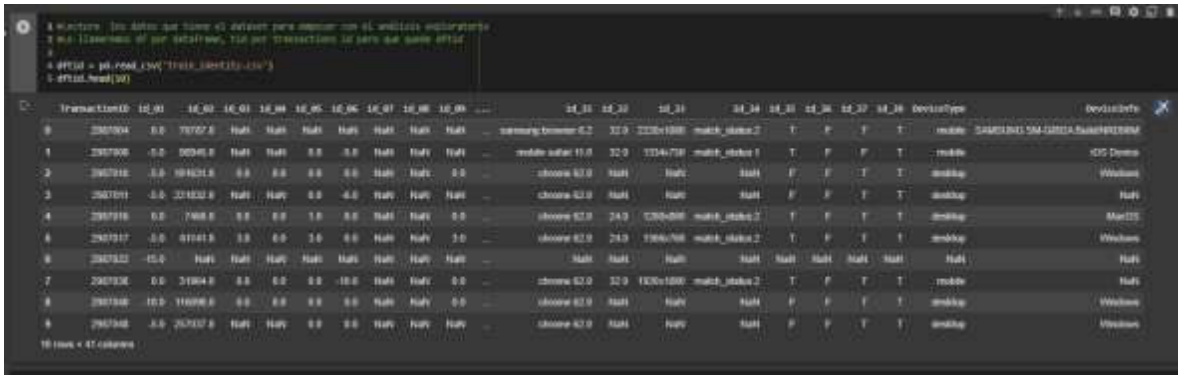
También podríamos determinar si los fraudes se realizaron más desde celulares o desde computadores y desde qué sistema operativo se realizaron.

Cuáles son los usuarios más vulnerados según el correo electrónico que usen, por ejemplo gmail, outlook o correos con dominios privados.

PROCESAMIENTO DE DATOS

Se realiza un preprocesamiento de datos identificando las columnas de ambas tablas. Para esto, se determinan cuáles son susceptibles para nuestras métricas.

Tenemos inicialmente dfi para el Data Frame de Identity

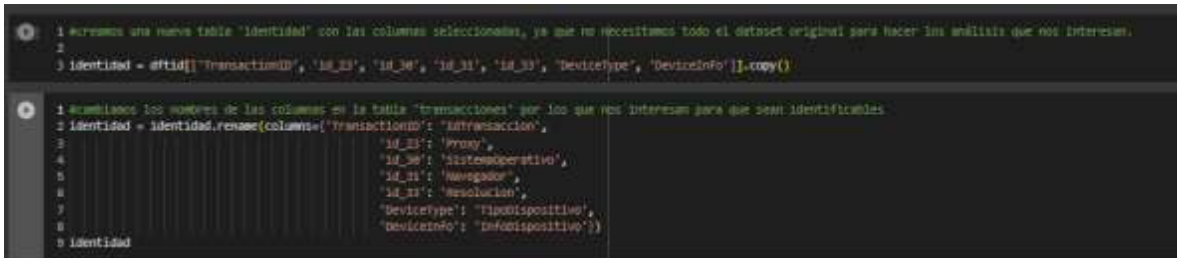


Para lo cual tenemos valores NaN con un porcentaje del 35.6%

```
Existen valores NaN: True
Total de valores NaN: 2104107
El porcentaje de valores NaN: 35.6%
```

Creamos un dftid_sinNan para filtrar todos los datos nulos y tener un dataframe más limpio.

Ahora cambiamos algunas cosas de Identidad así:



```
#hacemos los cambios pertinentes para Proxy
identidad['Proxy'] = identidad['Proxy'].replace(['IP_PROXY:TRANSPARENT'], 'Transparent')
identidad['Proxy'] = identidad['Proxy'].replace(['IP_PROXY:ANONYMOUS'], 'Anonymous')
identidad['Proxy'] = identidad['Proxy'].replace(['IP_PROXY:HIDDEN'], 'Hidden')
```

Más adelante podría interesarnos si los fraudes se realizan desde equipos detrás de proxies, con cual sistema operativo, desde que dispositivo, por lo cual son datos que se quedan a pesar de que son variables categóricas que luego se convertirán a numéricas.

Al tratar de hacer lo mismo con la tabla transacciones, nos damos cuenta que más del 41% son datos nulos y que no siempre coinciden con la clave primaria que es el ID de la transacción, por lo que decidimos dejarla tal cual está.

```
#Debemos verificar cuántos valores nulos tenemos en la tabla, para esto hacemos el chequeo asi:
check_for_any_nan= dfttr.isna().any().any()#Vamos a encontrar cuantos valores NaN existen en el dataframe
total_nan_values = dfttr.isna().sum().sum()#Vamos a sumar el total de valores NaN presentes en el dataframe
fil ,col = dfttr.shape
num_datos = fil*col
print("Existen valores NaN: "+str(check_for_any_nan))
print("Total de valores NaN: "+str(total_nan_values))
#print("El porcentaje de valores NaN: " +str((total_nan_values*100)/num_datos) + "%")
print("El porcentaje de valores NaN: {:.1f}%".format(round((total_nan_values*100)/num_datos, 1)))
```

```
Existen valores NaN: True
Total de valores NaN: 95566686
El porcentaje de valores NaN: 41.1%
```

Hacemos tratamiento de los valores faltantes:

```
transacciones.isnull().sum() #Columnas con valores nulos
```

```
IdTransaccion       0
EsFraude            0
LineaDeTiempo       0
MontoTransaccion    0
Franquicia          0
TipoTarjeta         0
DominioCompra       0
dtype: int64
```

```
[25]  transacciones['DominioCompra'] = transacciones['DominioCompra'].fillna('Sin Información')
      transacciones['Franquicia'] = transacciones['Franquicia'].fillna('Sin Información')
      transacciones['TipoTarjeta'] = transacciones['TipoTarjeta'].fillna('Sin Información')
```

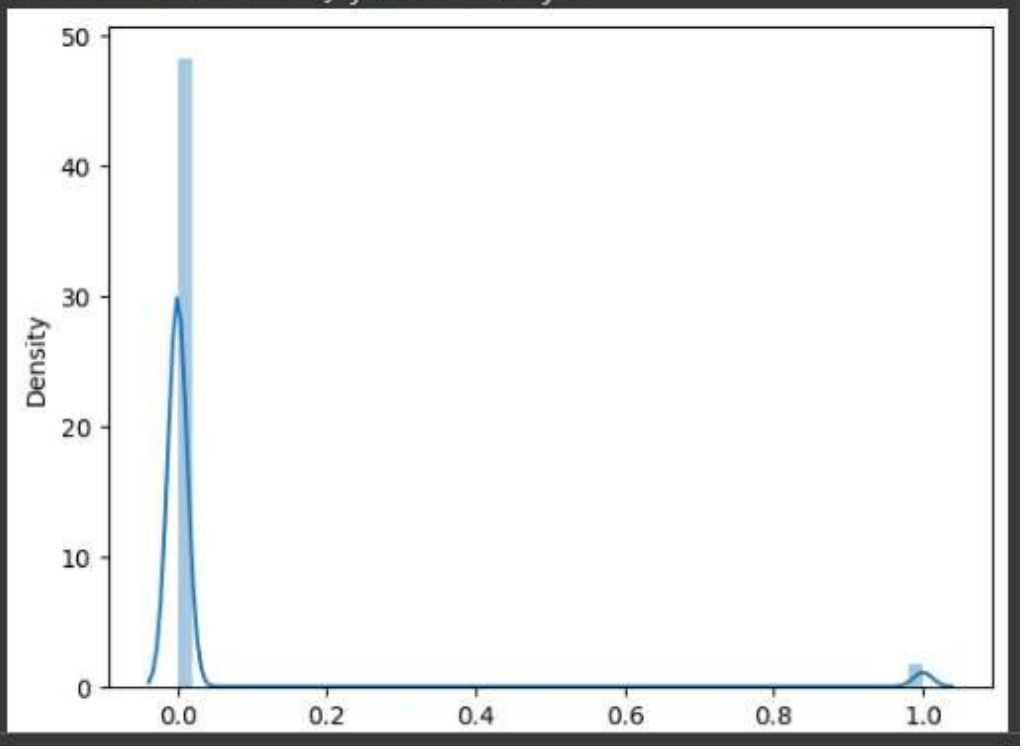
Rellenamos los nulos de df

```
[ ]  1 df.shape  #Tenemos 144.233 transacciones con 13 columnas que nos informan varias cosas sobre la transacción

     (144233, 13)

[28]  1 null_counts = df.isnull().sum()
      2 print(null_counts)
      3

     IdTransaccion            0
     Proxy               139064
     SistemaOperativo     66668
     Navegador             3951
     Resolucion           70944
     TipoDispositivo       3423
     InfoDispositivo      25567
     EsFraude                 0
     LineaDeTiempo            0
     MontoTransaccion         0
     Franquicia               0
     TipoTarjeta              0
     DominioCompra            0
     dtype: int64

      1 columns_to_fillna = ['Proxy', 'Navegador', 'Resolucion', 'TipoDispositivo', 'InfoDispositivo']
      2 for column in columns_to_fillna:
      3     df[column] = df[column].fillna('Sin Información')
```

ANALISIS DE LOS DATOS

Se analiza la variable objetivo IsFraud

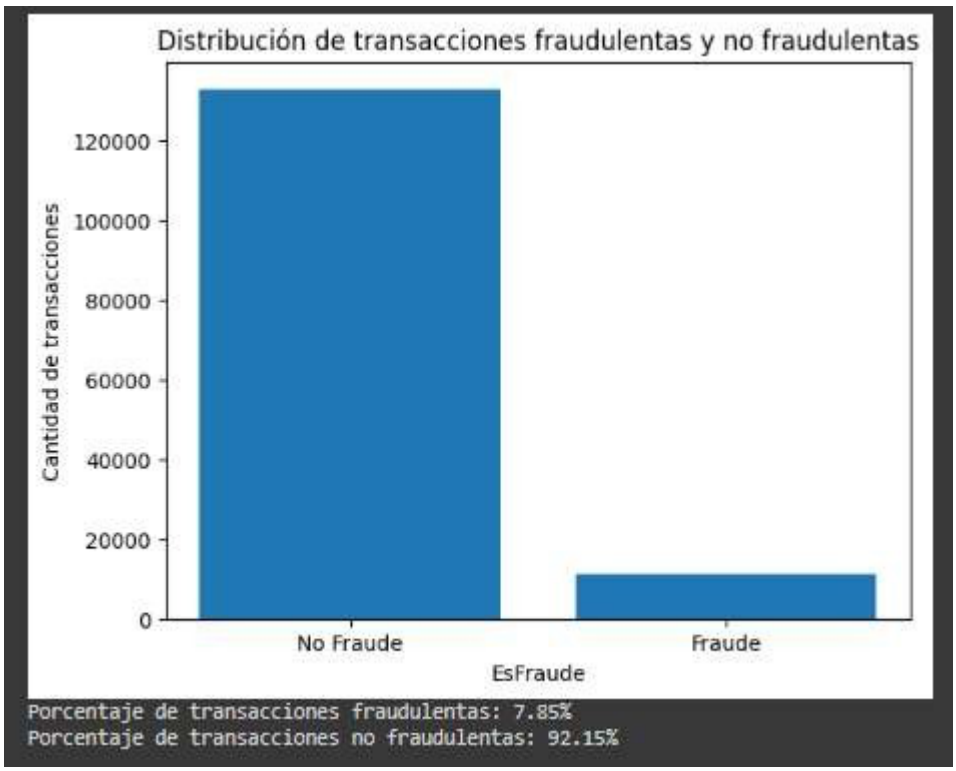
```
[ ]  1 #Distribución de la variable objetivo
     2 sns.distplot(df['isFraud'])

     <Axes: xlabel='isFraud', ylabel='Density'>
```



Y se mide su sesgo, donde se podría inferir que es una prueba de bondad y ajuste en la que la distribución obedece a una distribución normal.

```
[ ]  1 #Ahora mediremos el sesgo de nuestra variable objetivo
     2 print('Skewness de la variable objetivo {:.1f}%'.format(round((df['isFraud'].skew()),1)))

     Skewness de la variable objetivo 5.1%
```

Con esto podemos ver que nuestra variable objetivo EsFraude está desbalanceada
lo cual tiene toda la lógica ya que el porcentaje de fraudes siempre va a ser mucho menor de los que si lo son.



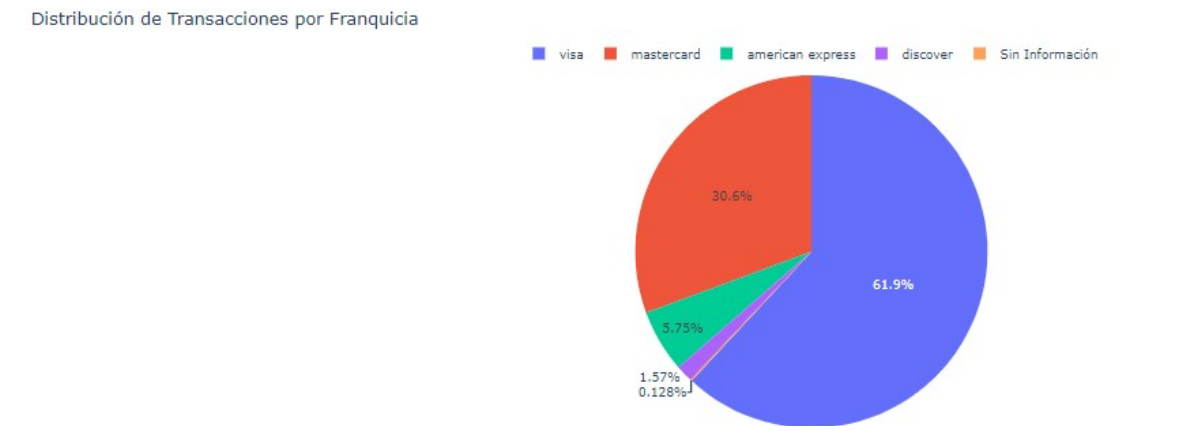
Se realizan algunas aproximaciones para entender un poco más los datos:

```
1 df.describe()
2
```

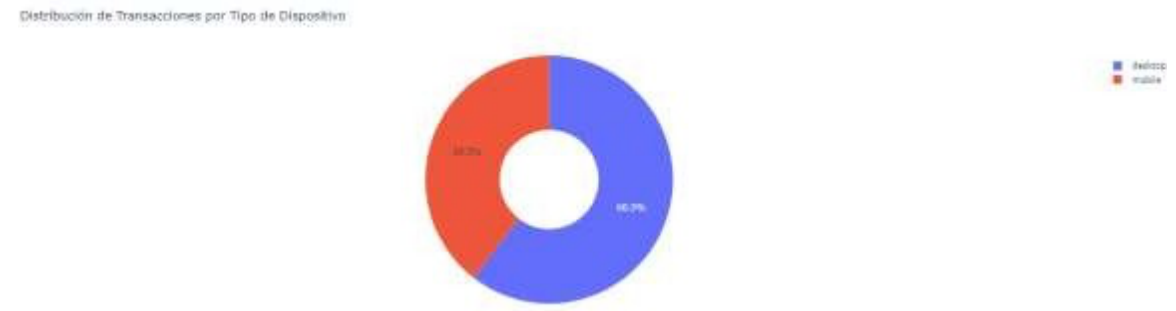
	IdTransaccion	EsFraude	LineaDeTiempo	MontoTransaccion
count	1.442330e+05	144233.000000	1.442330e+05	144233.000000
mean	3.236329e+06	0.078470	6.166958e+06	83.554533
std	1.788496e+05	0.268911	4.807714e+06	99.850258
min	2.987004e+06	0.000000	8.650600e+04	0.251000
25%	3.077142e+06	0.000000	1.885289e+06	25.453000
50%	3.198818e+06	0.000000	4.913738e+06	50.000000
75%	3.392923e+06	0.000000	1.025794e+07	100.000000
max	3.577534e+06	1.000000	1.581103e+07	1800.000000

Aquí se puede observar la distribución de los datos y su desbalance en las variables numéricas

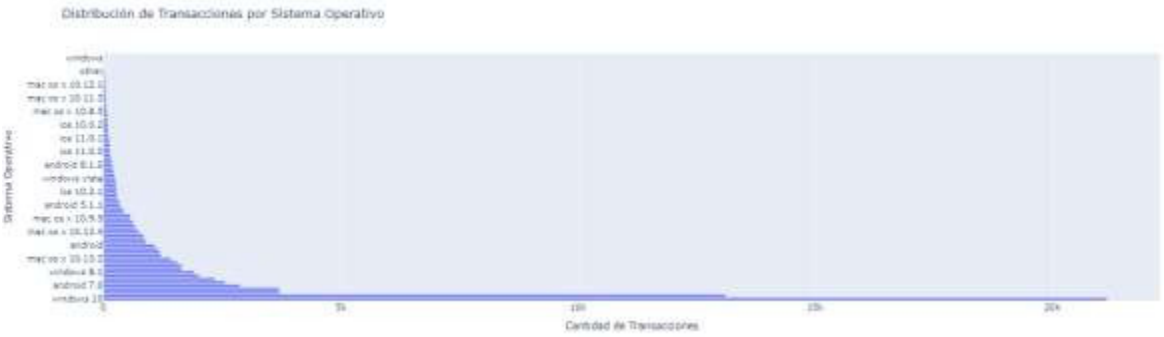
La distribución de transacciones por franquicia:



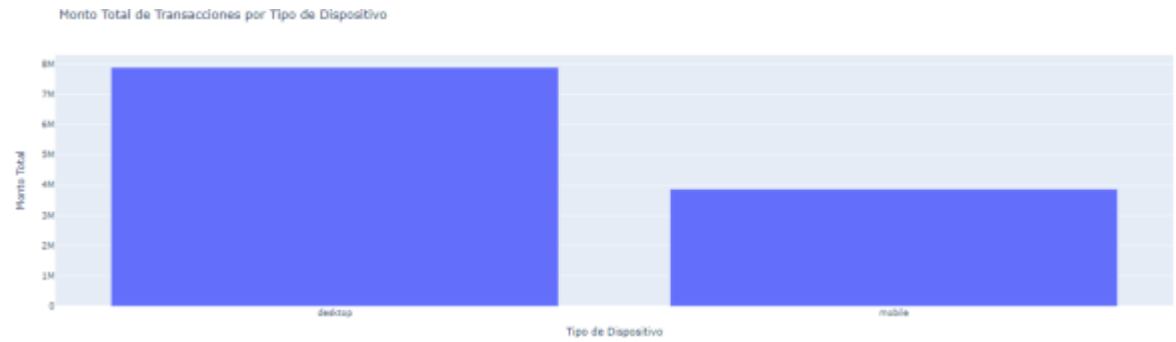
Por ejemplo, desde que dispositivo (móvil o escritorio) se realizan más fraudes:



Desde qué sistema operativo se hacen más fraudes:

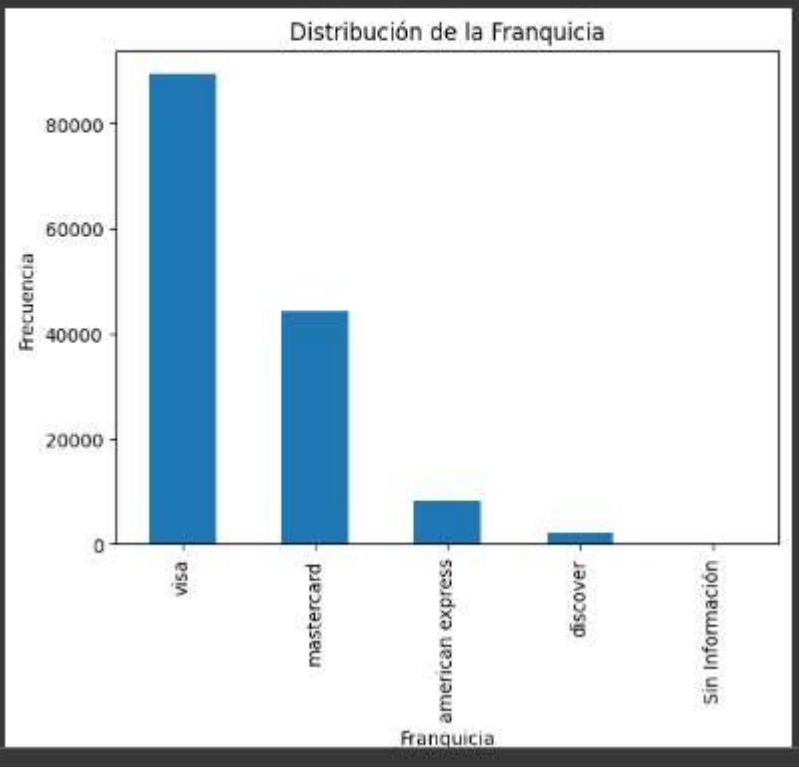


Transacciones por tipo de dispositivo

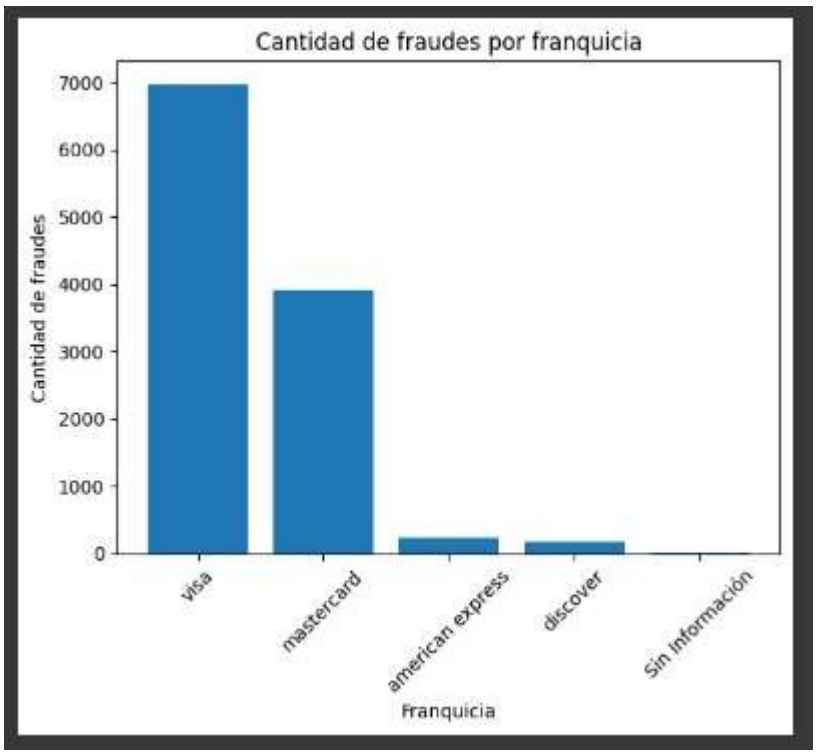


La distribución de las variables categóricas

```
[ ]  1 #Distribución de las variables categóricas
     2 df['Franquicia'].value_counts().plot(kind='bar')
     3 plt.xlabel('Franquicia')
     4 plt.ylabel('Frecuencia')
     5 plt.title('Distribución de la Franquicia')
     6 plt.show()
     7
```

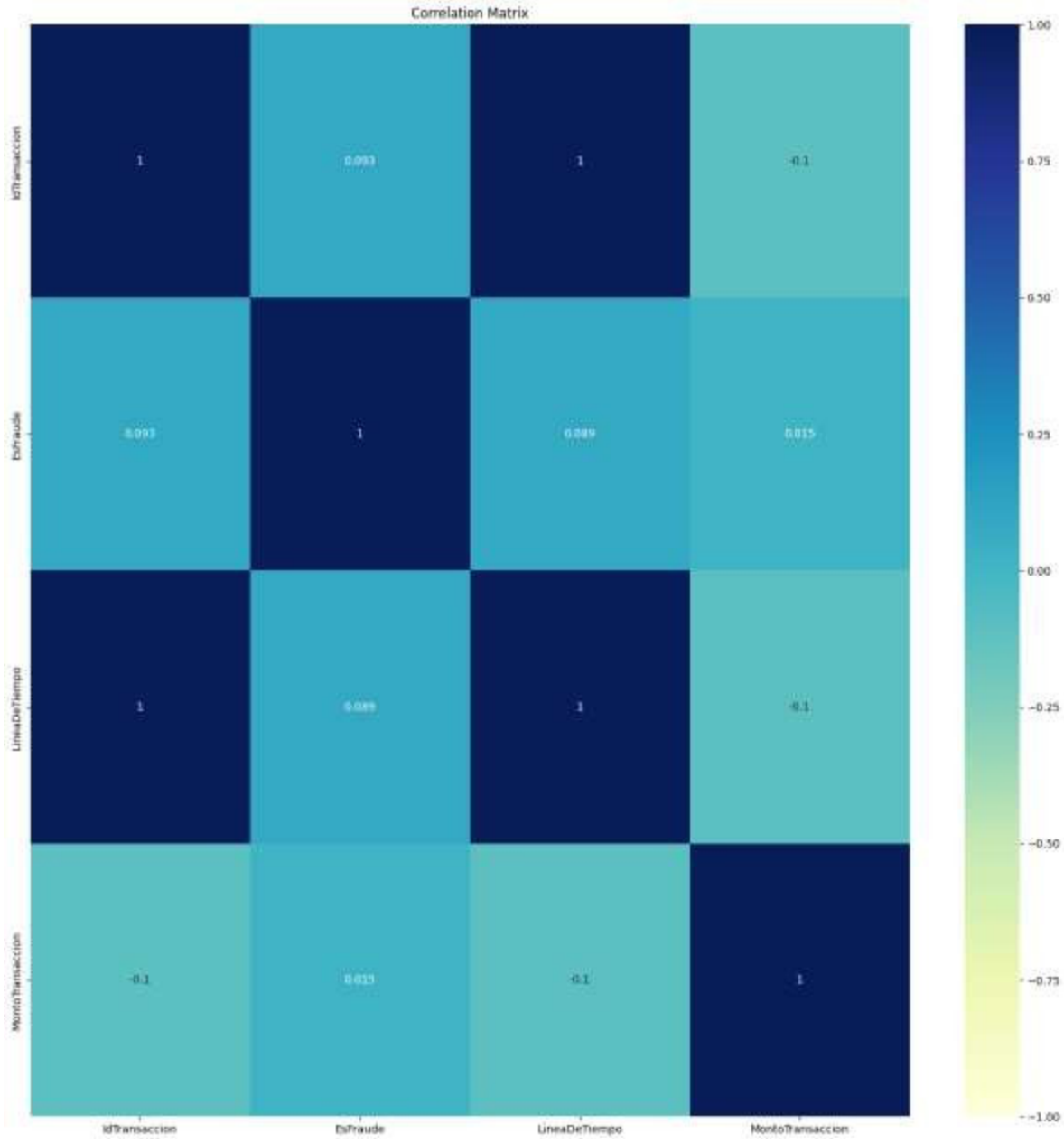


Cantidad de fraudes por franquicia

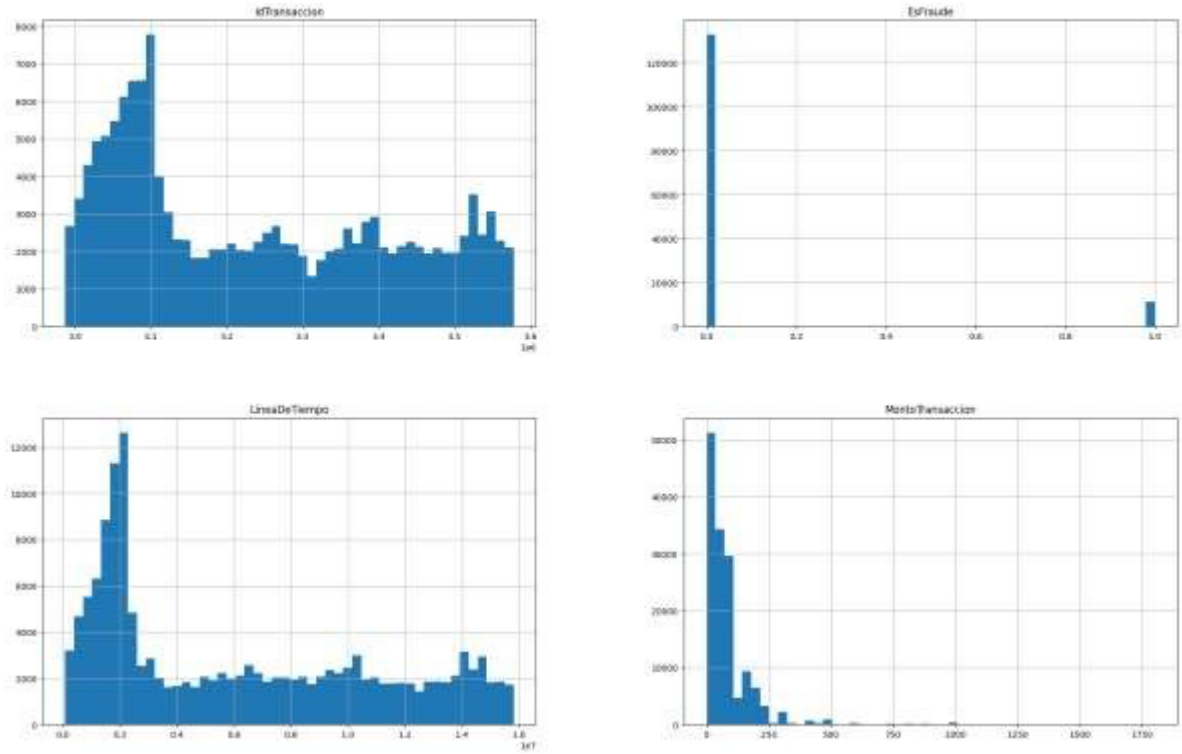


METRICAS DE EVALUACIÒN

Esta es nuestra matriz de correlación:



Y la distribución de las variables numéricas



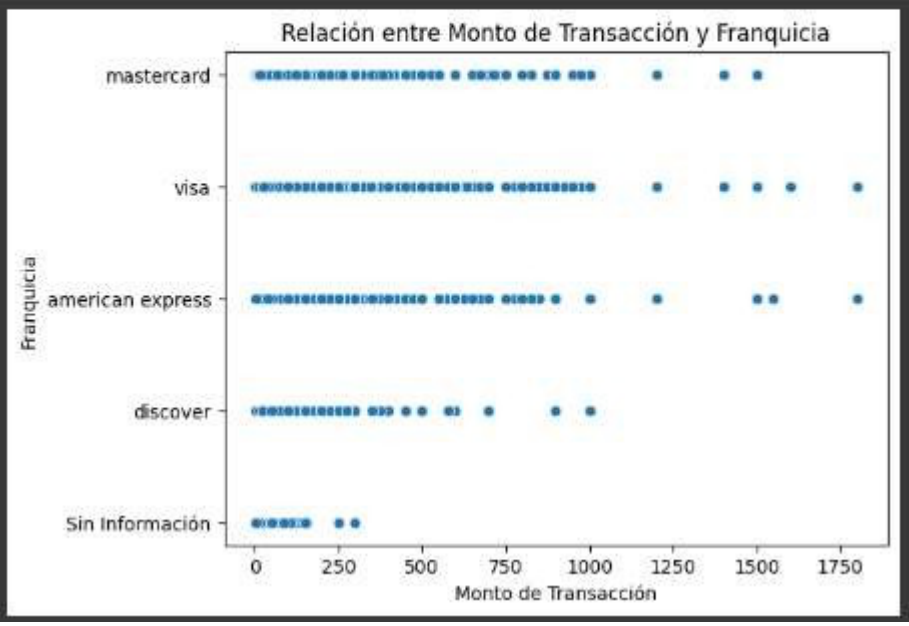
Nuestra matriz de correlaciones

```
1 #Correlaciones de las variables con la variable objetivo
2 target_corr_df = pd.DataFrame(corr_matrix["EsFraude"].sort_values(ascending=False))
3 target_corr_df
```

	EsFraude
EsFraude	1.000000
IdTransaccion	0.092935
LineaDeTiempo	0.089427
MontoTransaccion	0.015360

La relación entre el monto de la transacción y la franquicia

```
sns.scatterplot(x='MontoTransaccion', y='Franquicia', data=df)
plt.xlabel('Monto de Transacción')
plt.ylabel('Franquicia')
plt.title('Relación entre Monto de Transacción y Franquicia')
plt.show()
```



Los cuantiles teóricos

DIFICULTADES

Hemos encontrado que por ser un dataframe de datos bancarios, es información sensible, haciendo que Vesta no comparta demasiados datos que ellos usan para detectar fraude, limitando el dataset a solo algunos datos que permitan sacar conclusiones. Nos hubiera gustado datos más abiertos, por ejemplo, de ubicación. Si la persona estaba en un lugar diferente a su ubicación normal o desde que país se hacen más fraudes, pero esta información no está disponible.

CONCLUSIONES

Se puede concluir que la variable objetivo está desbalanceada debido a que el porcentaje de ningún fraude va hacer mayor que el de fraude desde la franquicia visa es donde se presentan más fraudes, con dispositivo desktop y con sistema operativo Windows 10. Según las relaciones de las métricas que se realizaron en el trabajo se puede identificar transacciones criticas por ejemplo, transacciones que se realicen desde móviles o desktop con la franquicia visa, esto con el fin de realizar controles antifraudes en estas situaciones, también se puede filtrar estos controles en horarios críticos como a media noche.