



Proyecto: Detector de fraudes en pagos vía online

Equipo 11. Integrantes:



- Ana Laura López --> any.lom.01@gmail.com
- Alma Violeta Alvarez --> almavioletaalvarez@gmail.com
- Ericka Daniela Cervantes --> erickadanielacervantesrios26@gmail.com



Planteamiento

Con el auge de las nuevas Tecnologías de Información y Comunicación, el uso de plataformas online para diferentes fines ha crecido exponencialmente. En relativamente poco tiempo pasamos de usar efectivo a usar tarjetas de crédito y débito.

Si bien estos cambios han facilitado las transacciones monetarias en el ámbito cibernético, las complicaciones de esta tecnología en expansión no se han hecho esperar, siendo una de estas las acciones de pago fraudulento en sus diversas modalidades



Objetivo

Desarrollar un proyecto que mediante el uso de algoritmos de ML para separar, validar y evaluar, así como de algoritmos supervisados como los árboles de decisión, permita identificar de manera eficiente las transacciones fraudulentas vía online.

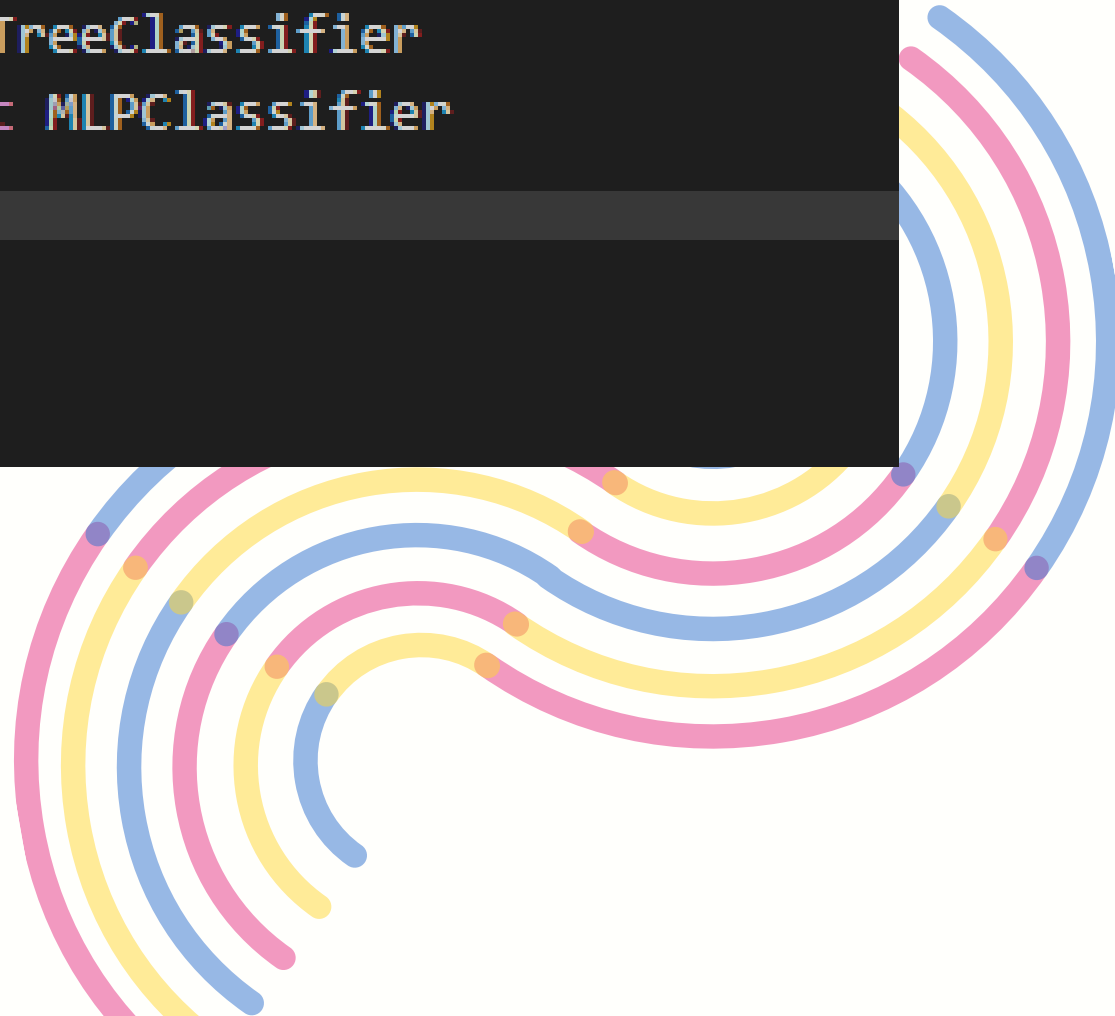
Esto con la intención de hacer del espacio online un entorno más seguro para las transacciones comerciales, lo cual se verá traducido en más y mejores avances comerciales que beneficien a oferentes y compradores.



Librerías utilizadas

```
[ ] import pandas as pd
import numpy as np
from scipy import stats
import seaborn as sns
from scipy.stats import kurtosis, skew
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
```

```
[ ] from google.colab import drive
drive.mount('/content/drive')
```



Análisis general de dataset

```
df1 = pd.read_csv('/content/drive/MyDrive/DataSetsAny/Payments-fraud.csv')  
df1.head(5)
```

	step	type	amount	nameOrig	oldbalanceOrig	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

```
[ ] #Dimensiones del dataset  
df1.shape  
  
(6362620, 11)
```

Dataset
sin datos
nulos

```
[ ] #Verificamos si tenemos valores nulos en nuestras columnas  
df1.isna().sum()
```

```
step          0  
type          0  
amount        0  
nameOrig      0  
oldbalanceOrig 0  
newbalanceOrig 0  
nameDest      0  
oldbalanceDest 0  
newbalanceDest 0  
isFraud        0  
isFlaggedFraud 0  
dtype: int64
```

Cambio de nombre en columnas

	hour_step	trans_type	amount	orig_client	orig_old_balance	orig_new_balance	dest_client	dest_old_balance	dest_new_balance	is_fraud	is_flagged_fraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Se realizó por convención para facilitar el análisis de variables numéricas y categóricas

Variable numerica: "amount"
Variable categórica "trans_type"

```
#Obtenemos estimados de locacion variable amount
print('Estimados de locacion')
print('amount')
print('Media:',df1['amount'].mean())
print('Mediana:',df1['amount'].median())
print('Media truncada:', stats.trim_mean(df1['amount'], 0.1))
print('Desv. estandar:', df1['amount'].std())
```

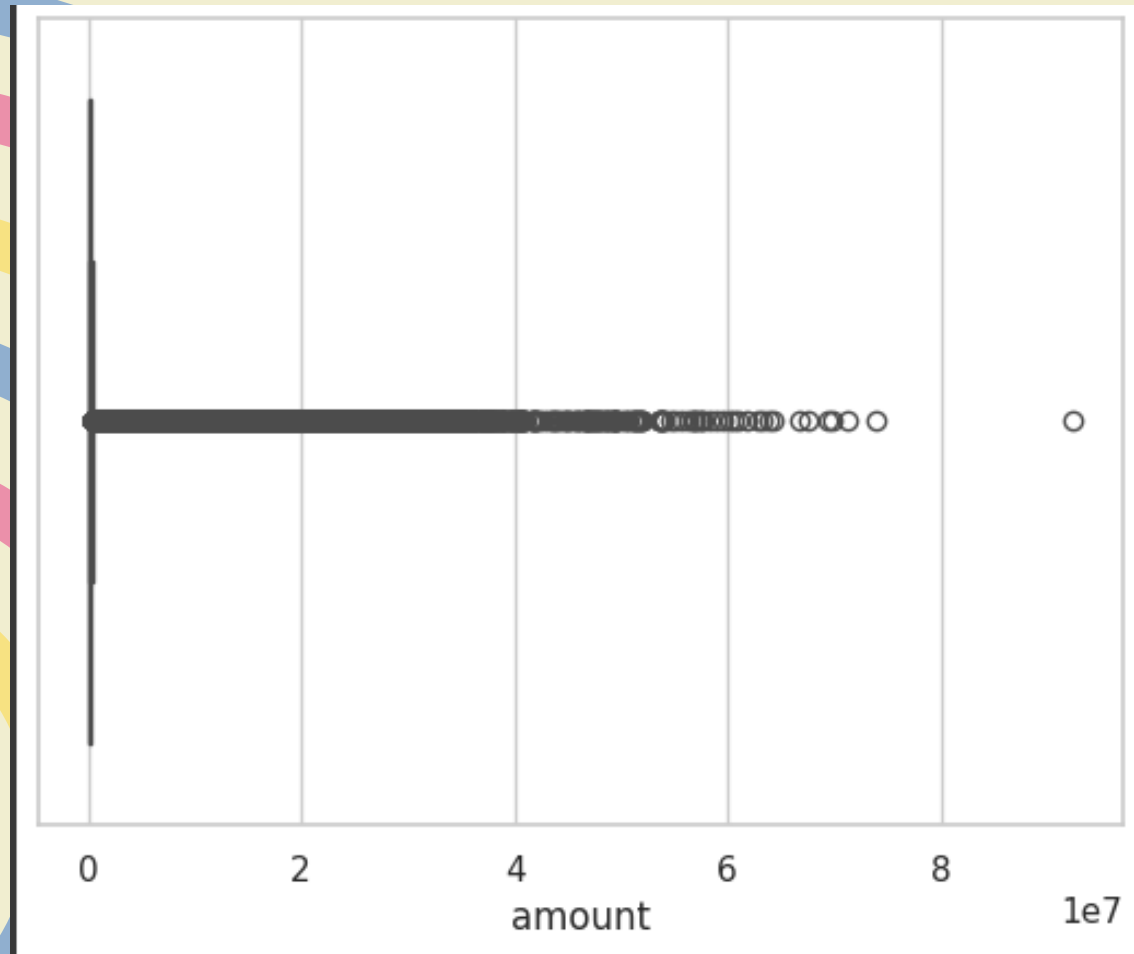
```
Estimados de locacion
amount
Media: 179861.90354913071
Mediana: 74871.94
Media truncada: 107007.98284970847
Desv. estandar: 603858.2314629358
```

```
[ ] # Obtenemos estimados de variabilidad
print('Estimados de variabilidad')
print('purchase_amount')
print('Valor minimo: ', df1['amount'].min())
print('Percentil 10:',df1['amount'].quantile(0.10))
print('Percentil 25:',df1['amount'].quantile(0.25))
print('Percentil 50:',df1['amount'].quantile(0.50))
print('Percentil 75:',df1['amount'].quantile(0.75))
print('Percentil 90:',df1['amount'].quantile(0.90))
print('Valor maximo:',df1['amount'].max())
```

```
Estimados de variabilidad
purchase_amount
Valor minimo: 0.0
Percentil 10: 4501.3
Percentil 25: 13389.57
Percentil 50: 74871.94
Percentil 75: 208721.4775
Percentil 90: 365423.309000000007
Valor maximo: 92445516.64
```

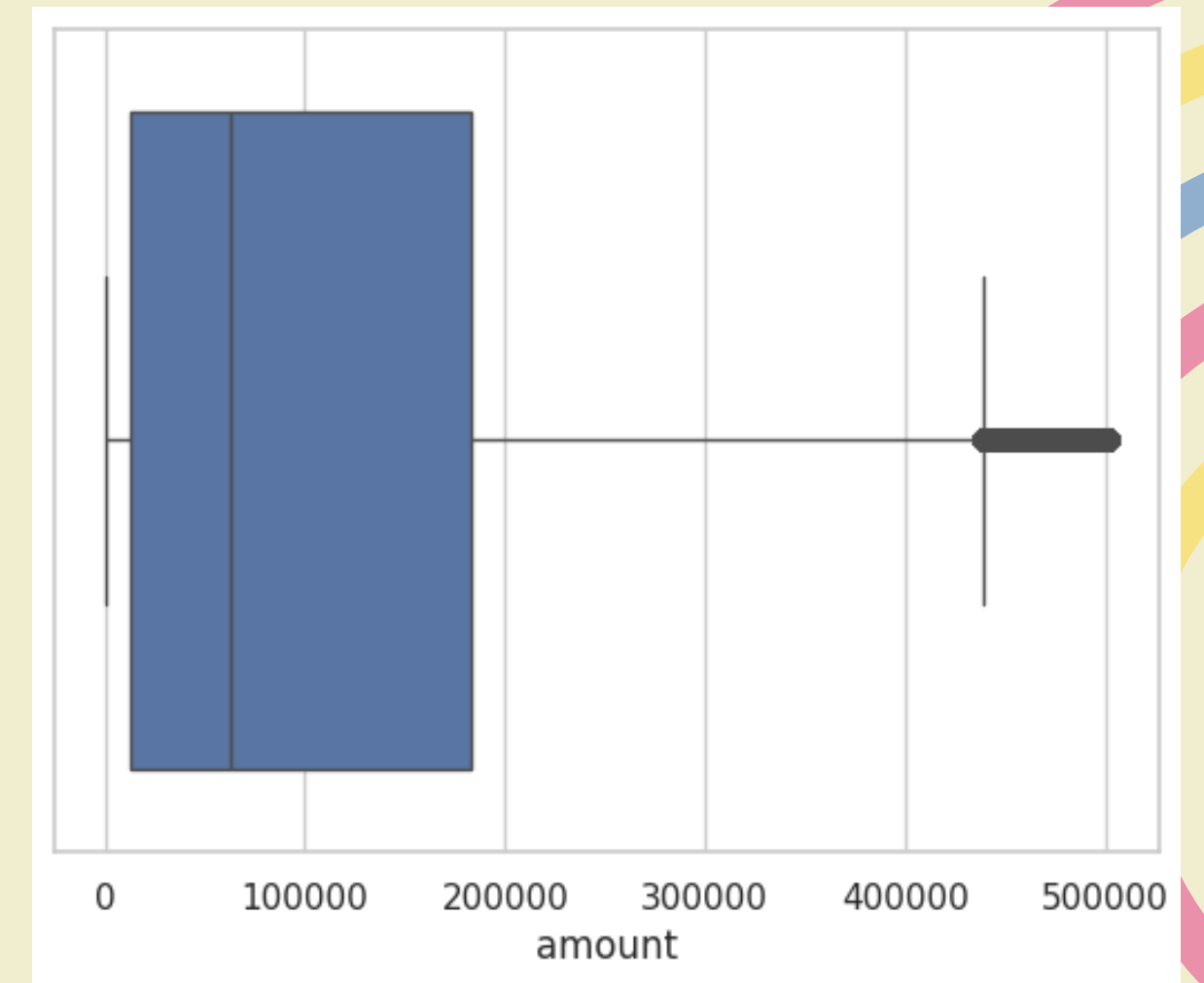
Percentil 50 a 75 con cambios relevantes en monto ---> Mayor acumulación de datos de montos altos = Media más alta que mediana.

Identificación y filtración de datos atípicos

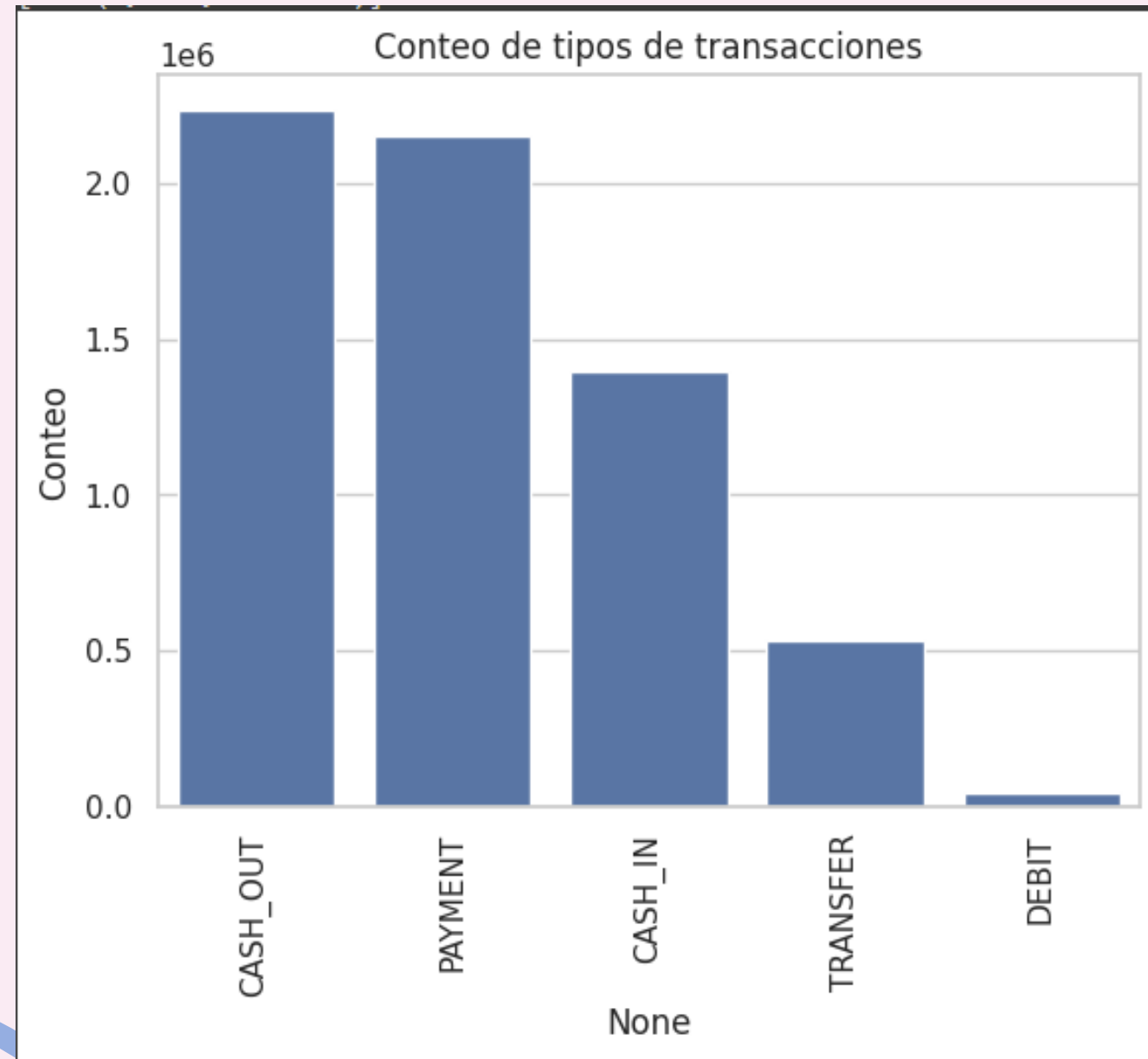


Antes

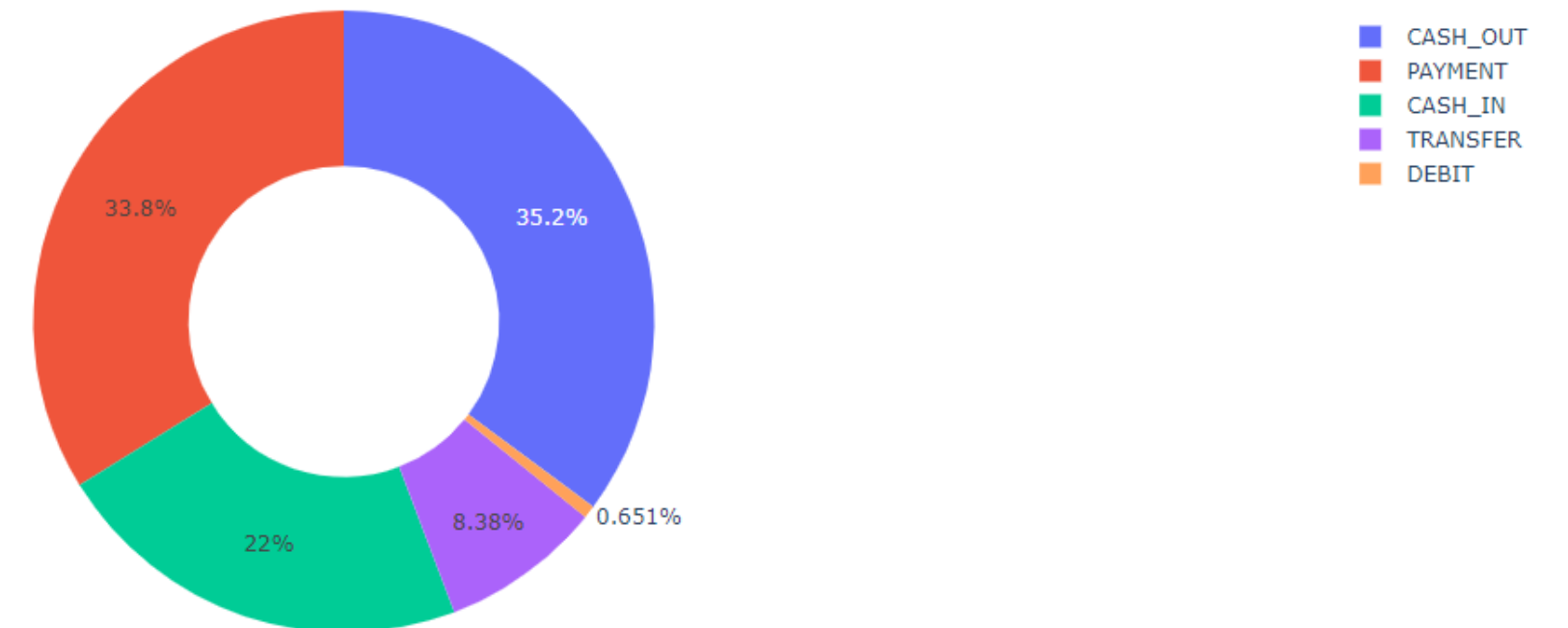
Después



Variable categórica potencial= transacción



Representación de mayor proporción en movimientos de retiros y pagos.



Correlación entre variables

	hour_step	amount	orig_old_balance	orig_new_balance	dest_old_balance	dest_new_balance	is_fraud	is_flagged_fraud
hour_step	1.000000	0.022373	-0.010058	-0.010299	0.027665	0.025888	0.031578	0.003277
amount	0.022373	1.000000	-0.002762	-0.007861	0.294137	0.459304	0.076688	0.012295
orig_old_balance	-0.010058	-0.002762	1.000000	0.998803	0.066243	0.042029	0.010154	0.003835
orig_new_balance	-0.010299	-0.007861	0.998803	1.000000	0.067812	0.041837	-0.008148	0.003776
dest_old_balance	0.027665	0.294137	0.066243	0.067812	1.000000	0.976569	-0.005885	-0.000513
dest_new_balance	0.025888	0.459304	0.042029	0.041837	0.976569	1.000000	0.000535	-0.000529
is_fraud	0.031578	0.076688	0.010154	-0.008148	-0.005885	0.000535	1.000000	0.044109
is_flagged_fraud	0.003277	0.012295	0.003835	0.003776	-0.000513	-0.000529	0.044109	1.000000

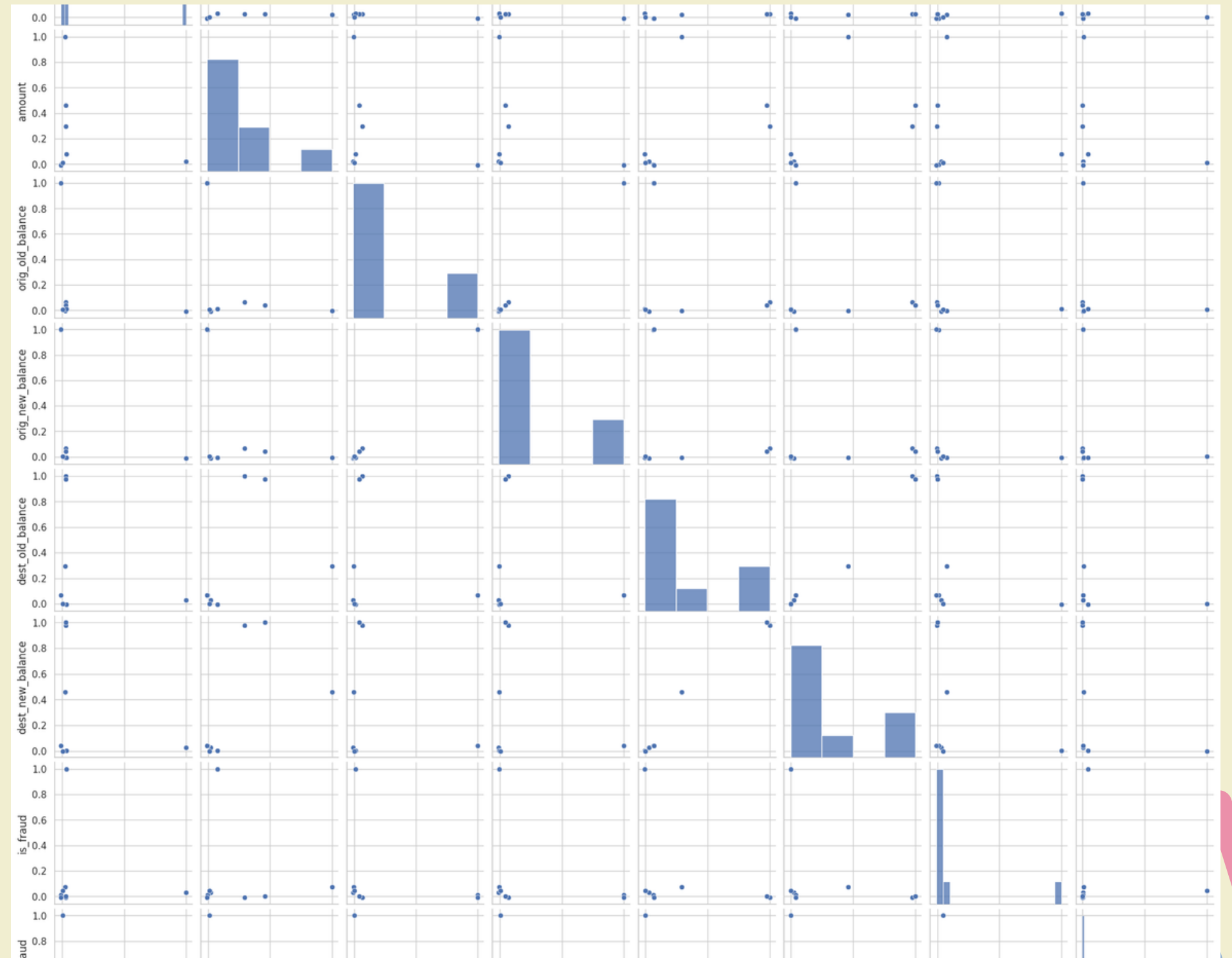
Hemos obtenido dos pares de variables que tienen un alto valor de correlación las cuales son:

- **orig_new_balance || orig_old_balance**
- **dest_old_balance || dest_new_balance**

Correlación

Son referidas al saldo de la cuenta antes y después de realizar una transacción, tanto del mismo usuario como el usuario destinatario involucrado.

Serán estas variables las utilizadas para nuestro algoritmo de **Machine Learning**.





Transformar las variables categóricas en numéricas

```
#Transformar las variables categoricas en numéricas
df1['trans_type'] = df1['trans_type'].map( {
    'CASH_OUT': 1,
    'PAYMENT' : 2,
    'CASH_IN': 3,
    'TRANSFER': 4,
    'DEBIT': 5
})
```

Cada tipo de transacción le asignamos un numero del 1 al 5, con el cual serán etiquetadas.



Variable dependiente

```
[ ] df1['is_fraud'] = df1['is_fraud'].map({  
    0: 'No Fraud',  
    1: 'Fraud'  
})
```

Nuestra variable dependiente, que en este caso es "is_fraud", que ya cuenta con las etiquetas de los registros de nuestro dataset, que nos ayudara a identificar futuros casos, será categorizada como

- 0: No fraude
- 1: Fraude

Los cambios realizados a nuestros datos

	hour_step	trans_type	amount	orig_client	orig_old_balance	orig_new_balance	dest_client	dest_old_balance	dest_new_balance	is_fraud	is_flagged
0	1	2	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	No Fraud	
1	1	2	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	No Fraud	
2	1	4	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	Fraud	
3	1	1	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	Fraud	
4	1	2	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	No Fraud	

Machine Learning

Comenzamos con el proceso para particionar nuestros datos.

Siguiendo la teoría vista en clases decidimos hacer una partición de la siguiente manera:

- Entrenamiento: 60%
- Validación: 10%
- Pruebas: 30%

variables independientes

```
[ ] #Asignamos los resultados de nuestra función a nuestras variables correspondientes
[x_train, x_val, x_test, y_train, y_val, y_test] = particionar(x,y,0.60,0.10,0.30)

print(x_train.shape)
print(x_val.shape)
print(x_test.shape)
```

0.4
(3817572, 4)
(636262, 4)
(1908786, 4)

- trans_type
- amount
- orig_old_balance
- orig_new_balance

Método de regresión lineal

Primero se intentó entrenar nuestro datl método de regresión lineal, sin embargo este no fue exitoso, debido a que nuestros datos estan etiquetados con una variable que solo tiene dos posibilidades,

```
from sklearn.metrics import confusion_matrix

def calcularAccuracy(TP, TN, FP, FN):
    accuracy = (TP + TN) / (TP + TN + FP + FN)
    accuracy = accuracy * 100
    return accuracy

def calcularSensibilidad(TP, TN, FP, FN):
    sensibilidad = TP / (TP + FN)
    sensibilidad = sensibilidad * 100
    return sensibilidad

def calcularEspecificidad(TP, TN, FP, FN):
    especificidad = TN / (TN + FP)
    especificidad = especificidad * 100
    return especificidad
```

PRIMER MODELO: Árbol de decisiones

```
[ ] #Primer modelo: ARBOL DE DECISIONES
ModelTree = DecisionTreeClassifier()
ModelTree.fit(x_train, y_train)
y_pred = ModelTree.predict(x_test)
print(ModelTree.score(x_test, y_test))
```

```
0.9996652322470931
```

```
[ ] resultado = confusion_matrix(y_test, y_pred)
print(resultado)
(TN, FP, FN, TP) = resultado.ravel()
print("True negatives: "+str(TN))
print("False positives: "+str(FP))
print("False negative: "+str(FN))
print("True positives: "+str(TP))
```

```
acc = calcularAccuracy(TP, TN, FP, FN)
sen = calcularSensibilidad(TP, TN, FP, FN)
spec = calcularEspecificidad(TP, TN, FP, FN)
```

Nuestro algoritmo cuenta con 2 etiquetas, entonces podría considerarse para ser un clasificador y un regresor, pero en este caso no lineal.

Las medidas de error de nuestro modelo

```
[[ 2161  355]
 [ 284 1905986]]
True negatives: 2161
False positives: 355
False negative: 284
True positives: 1905986
Precision:99.96652322470932%
Sensibilidad:99.98510179565329%
Especificidad:85.89030206677265%
```

Obtenemos las medidas de error de nuestro modelo, las cuales son consideradas muy buenas. La medida más baja obtenida fue la de especificidad, y con este resultado podemos concluir que nuestro modelo falló más en clasificar los valores negativos.

SEGUNDO MODELO: Red Neuronal

```
[ ] redNeuronal = MLPClassifier(hidden_layer_sizes=(4,2),  
                                max_iter=20000,  
                                activation = 'logistic',  
                                tol= 1e-7)
```

```
redNeuronal.fit(x_train, y_train)  
y_pred_neural = redNeuronal.predict(x_test)  
print(redNeuronal.score(x_test, y_test))  
print("Entrenamiento completado!")
```

```
0.9986818847162542  
Entrenamiento completado!
```

Al usar la una red neuronal como lo mencionamos anteriormente podemos evitar minimos locales.

Medidas de error de precisión y sensibilidad


Nuestras medidas de error de precisión y sensibilidad fueron muy altas, sin embargo nuestra medida de especificidad tuvo un gran fallo en la medida de especificidad.

```
[[      0      2516]
 [      0 1906270]]
True negatives: 0
False positives: 2516
False negative: 0
True positives: 1906270
Precision:99.86818847162542%
Sensibilidad:100.0%
Especificidad:0.0%
```



Método de procesamiento de imágenes

No pudimos utilizar el método de procesamiento de imágenes, debido a que nuestro dataset no era adecuado para este método. Sin embargo, podemos concluir que el método que resultó más idóneo para nuestro dataset, fue el de árboles de decisión porque nos entregó mejores resultados tanto en score como en las medidas de error.



**GRACIAS
POR SU AMABLE
Y RESPETUOSA
ATENCIÓN**

