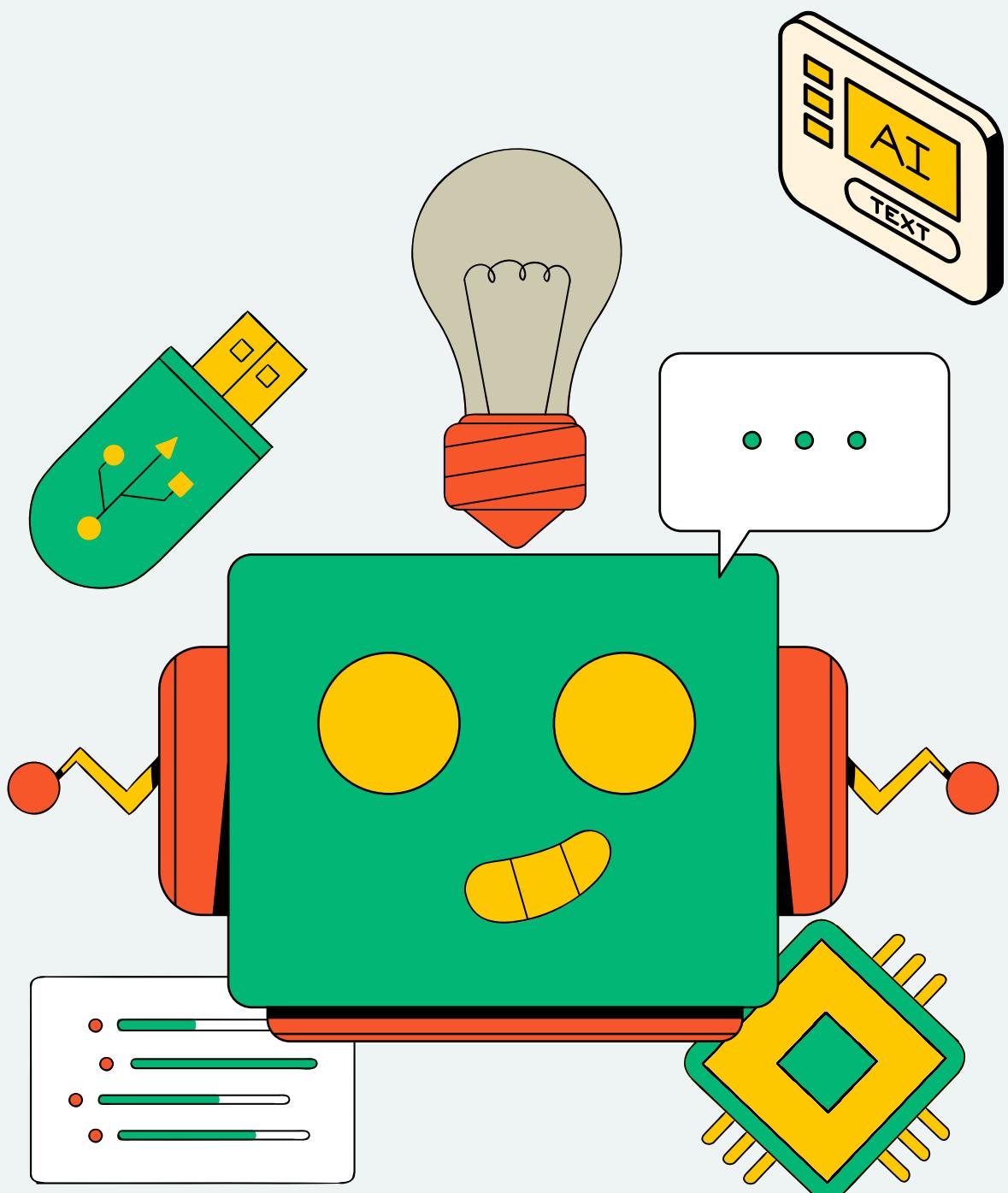




PRUEBA TÉCNICA
MERCADO LIBRE

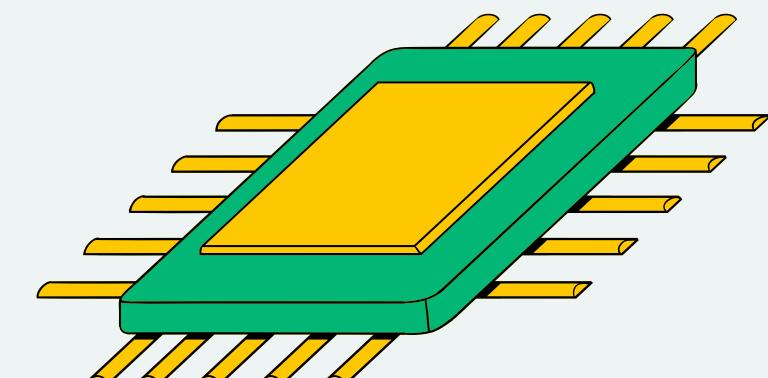


ENTREGA FINAL MACHINE LEARNING: DETECCIÓN DE FRAUDE

PRESENTACIÓN

REALIZADA POR

ELIZABETH GRANDA



OUTLINE DE LA PRESENTACIÓN

- ¿Qué problema estamos resolviendo?
- Una visión de los datos: EDA
- ¿Qué relaciones encontramos?
- Empezando el modelo
- Feature Engineering
- Features escogidas
- Modelos probados, modelos escogidos
- Deploy del modelo



¿QUÉ ESTAMOS RESOLVIENDO?

Para la instancia evaluativa final del curso de Machine Learning se pidió realizar un modelo que pudiera detectar y/o clasificar de la mejor forma las transacciones fraudulentas que se llevaron a cabo en diferentes países a través de la plataforma Mercado Libre



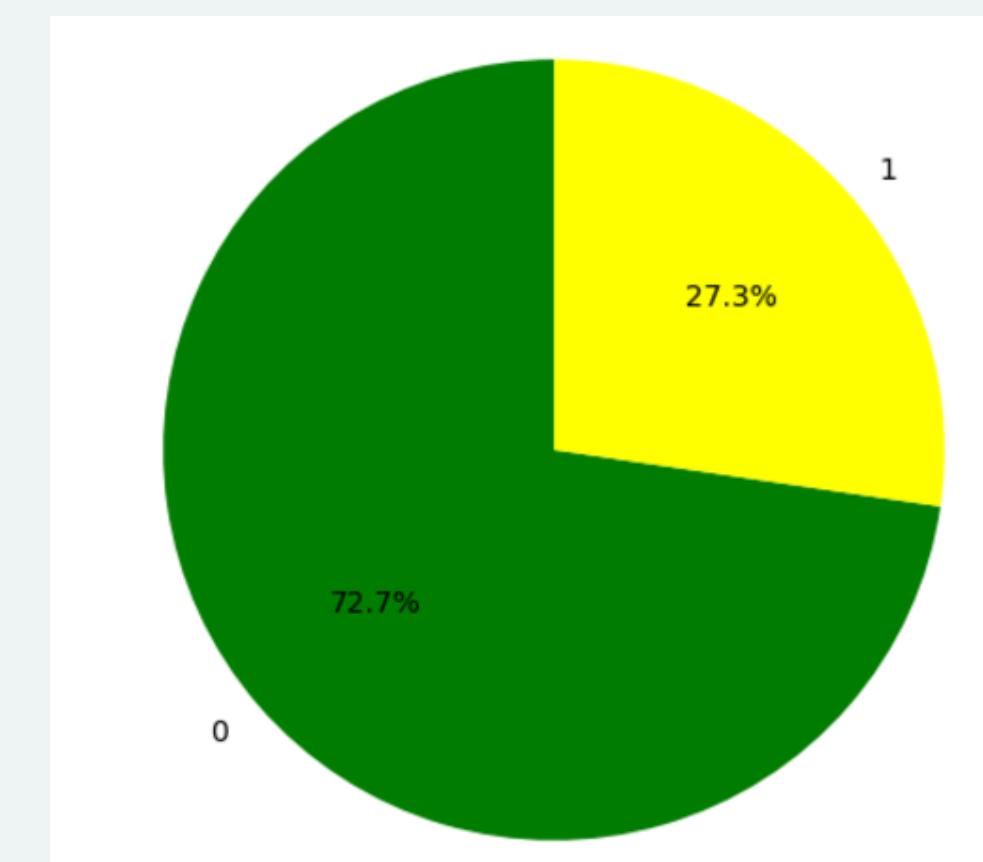
A	B	C	D	E	F	G	H	I	J	...	L	M	N	O	P	Q	R	S	Monto	Fraude	
0	0	10	50257.0	0	0	0.0	0.0	0	0	UY	...	0	3	1	0	5	0.00	0.00	7.25	37.51	1
1	0	10	29014.0	0	0	0.0	0.0	0	0	UY	...	0	1	1	0	3	0.00	0.00	11.66	8.18	1
2	0	7	92.0	0	1	0.0	0.0	0	1	UY	...	0	3	1	0	2	0.00	0.00	86.97	13.96	1
3	9	16	50269.0	0	0	0.0	0.0	0	0	UY	...	0	3	1	0	5	0.00	0.00	2.51	93.67	1
4	0	8	8180.0	0	0	0.0	0.0	0	0	UY	...	0	1	1	0	1	0.00	0.00	25.96	135.40	1

El dataset contiene 16880 observaciones y 21 columnas las cuales están nombradas con letras del abecedario y de la que no se conoce contexto alguno excepto por el Monto de la transacción y el país de origen.



UNA VISIÓN DE LOS DATOS: EDA

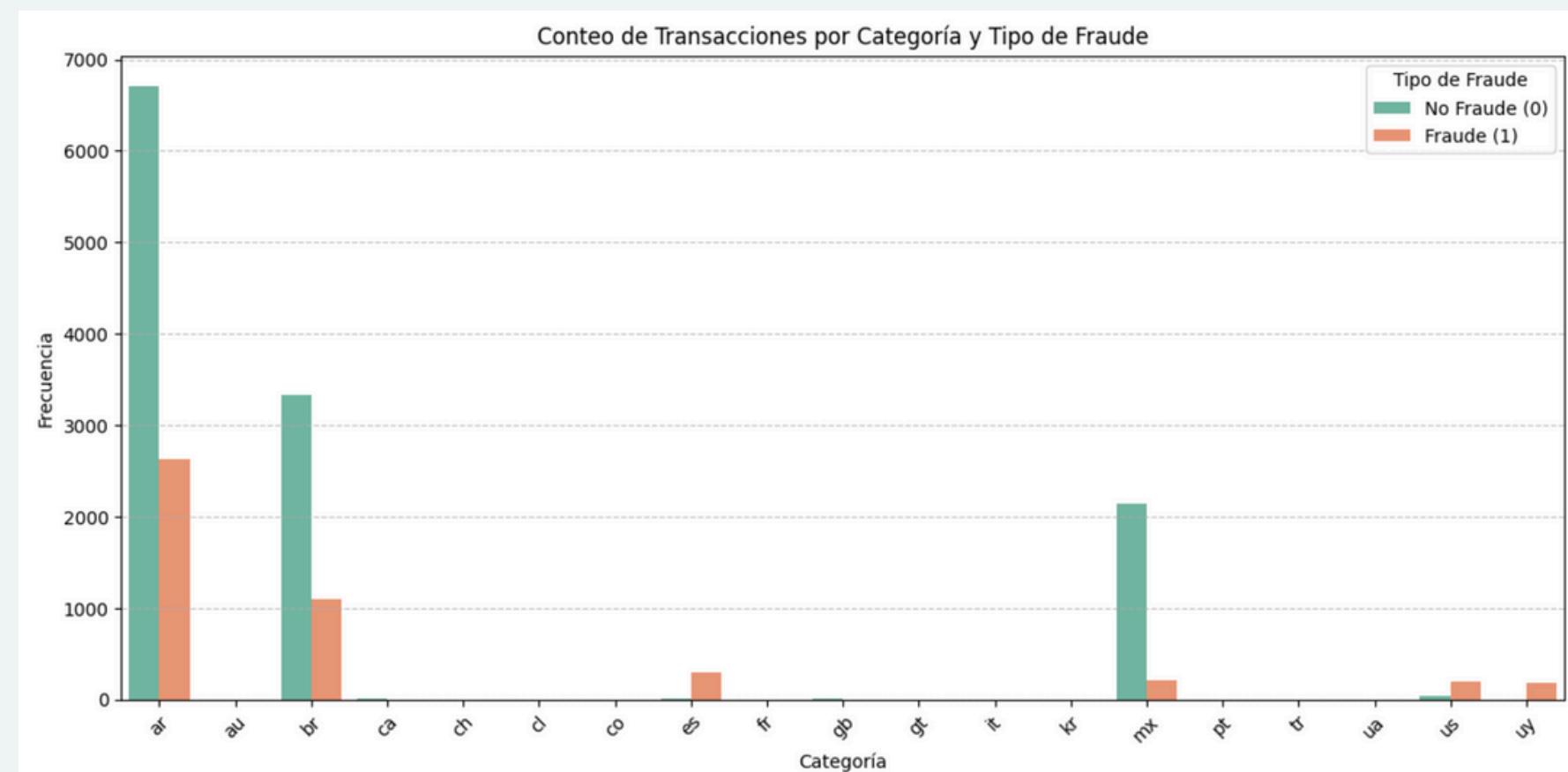
Con el fin de realizar un análisis inicial de los datos, se realizó un Exploratory Data Analysis con el fin de descubrir insights y/o patrones entre las variables y su relación con el target fraude



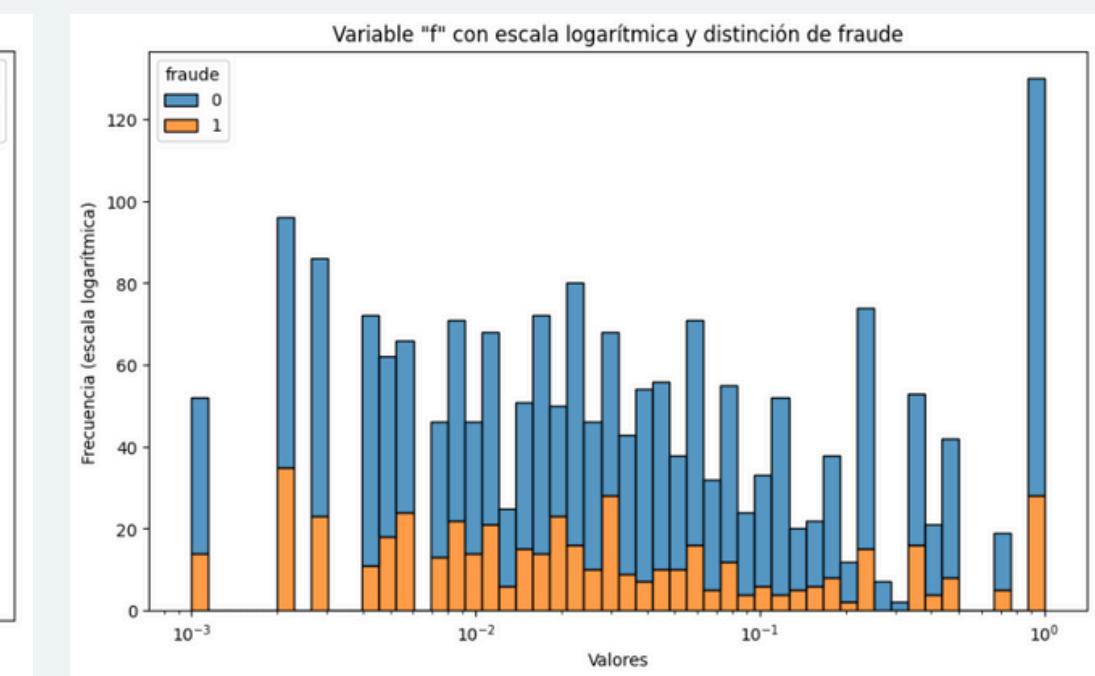
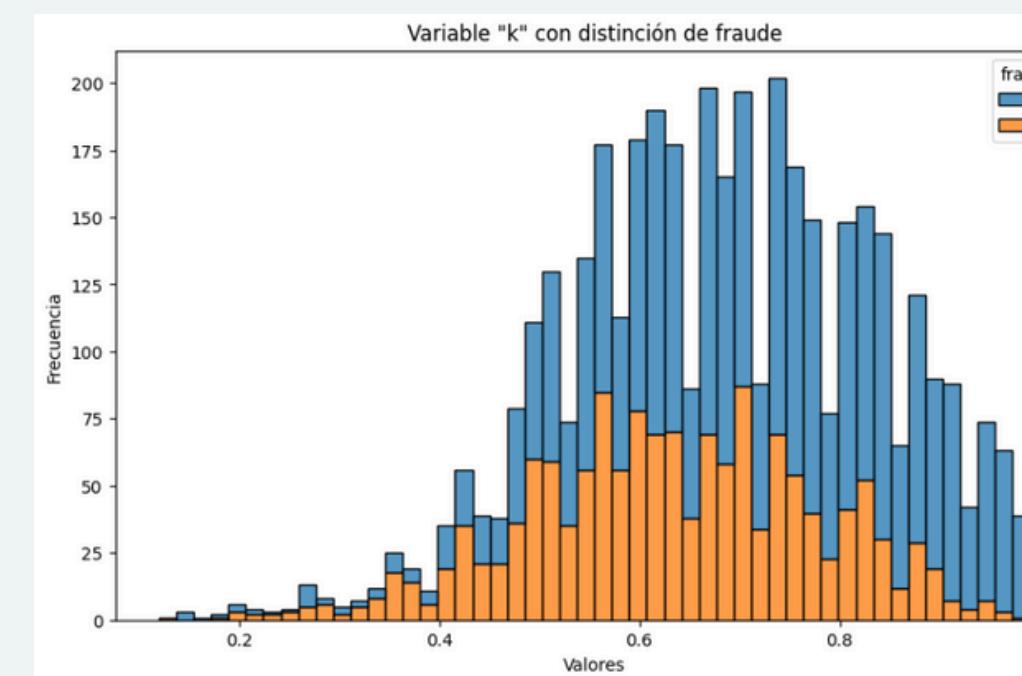
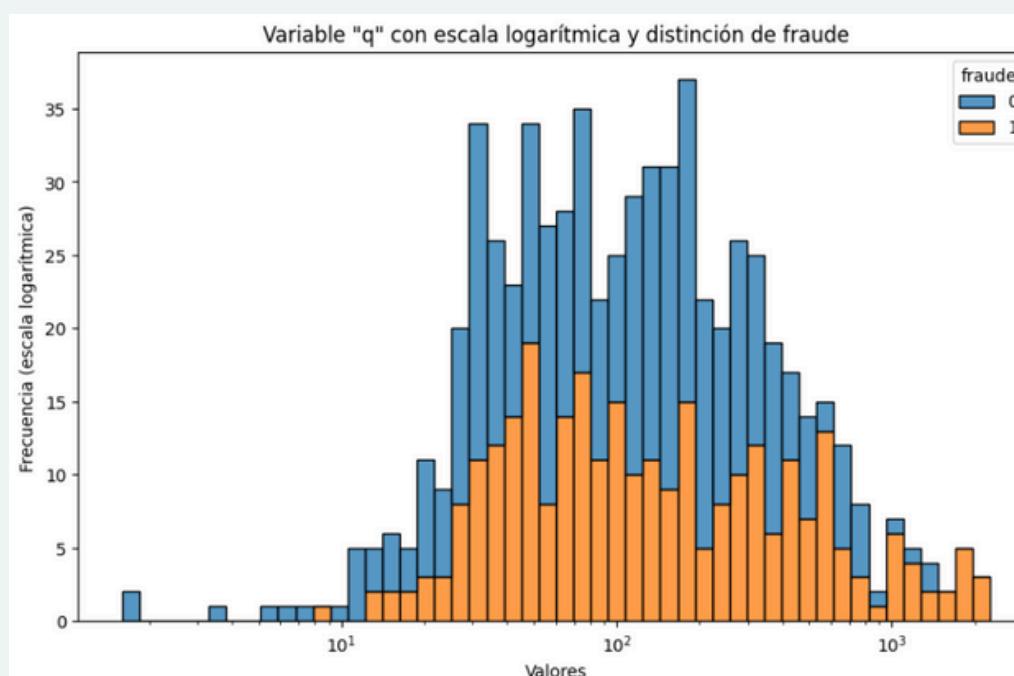
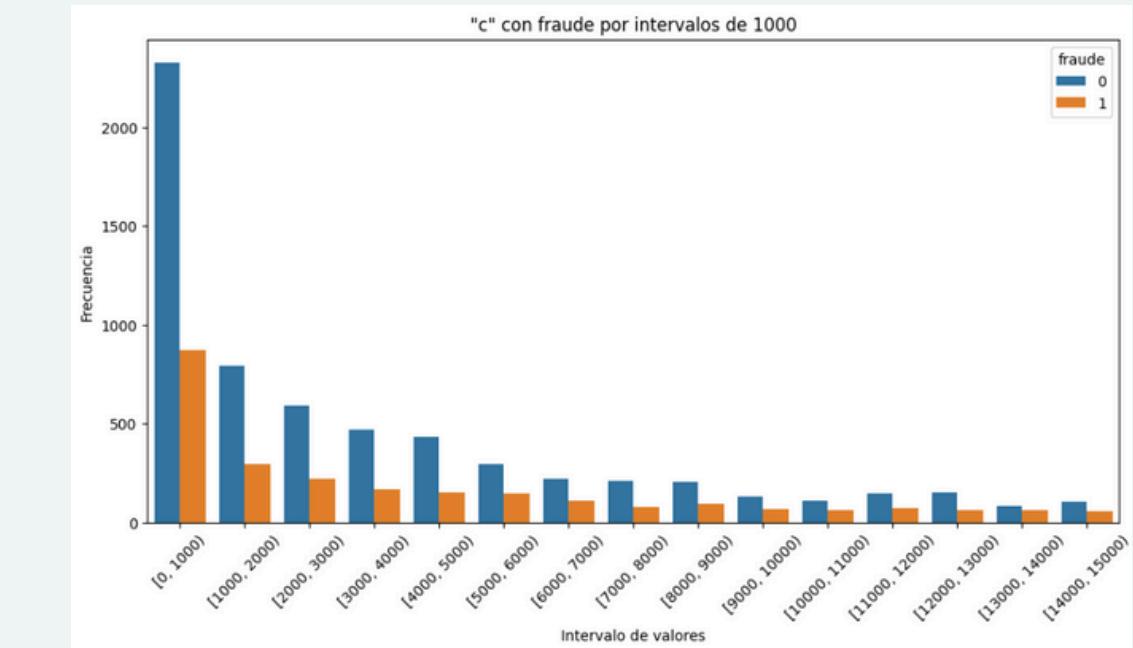
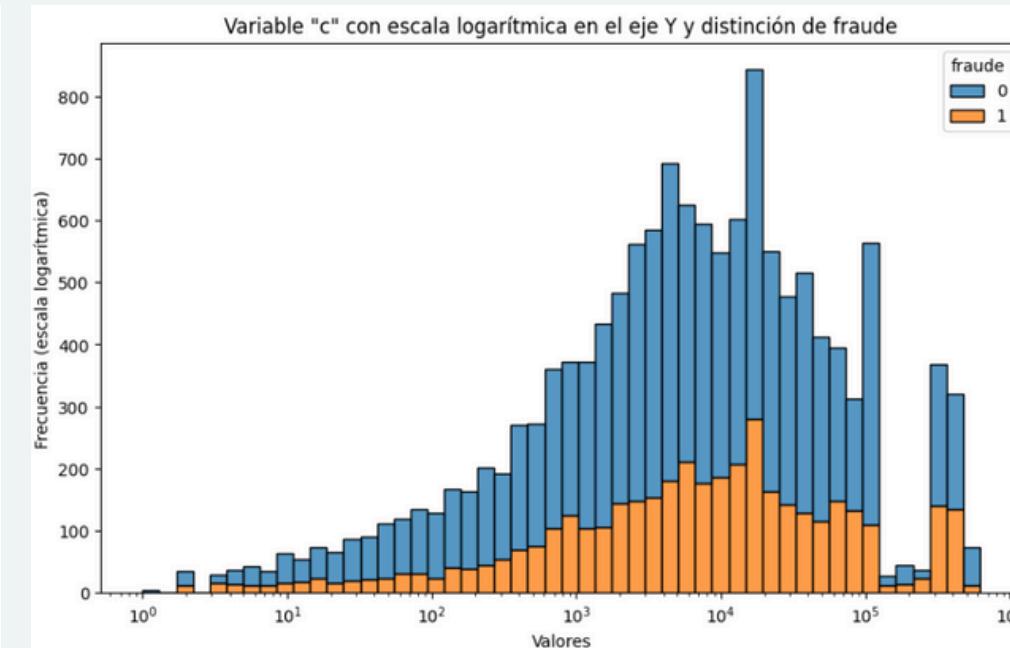
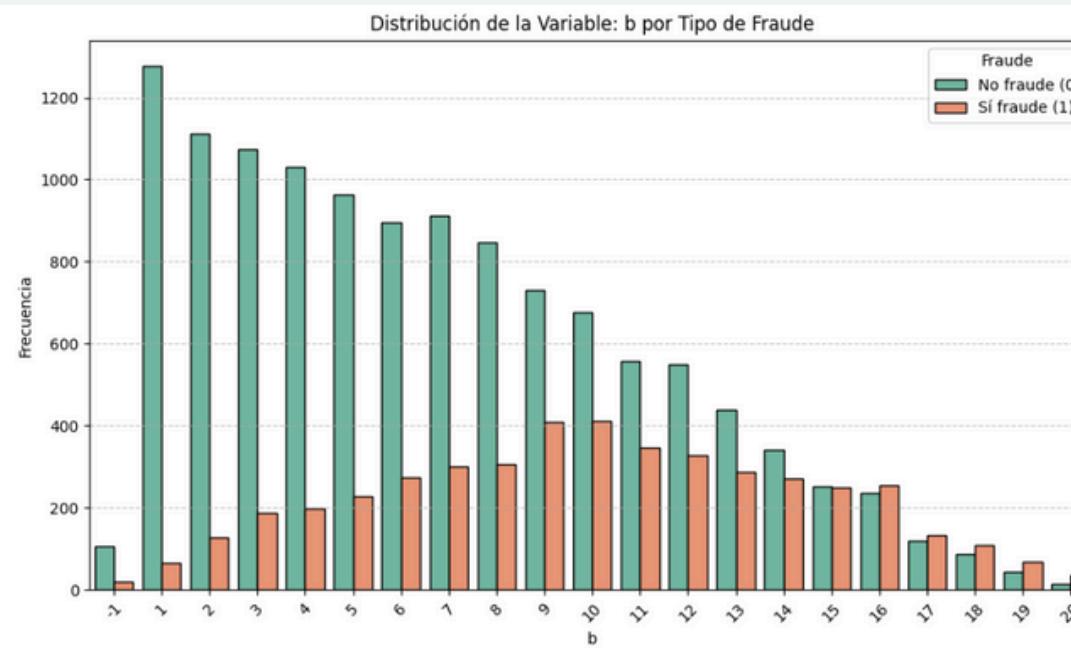
Se observa inicialmente la proporción de lo que es fraude y lo que no es: la data está desbalanceada, el modelo a aplicar debe tener esto en cuenta para no caer en errores



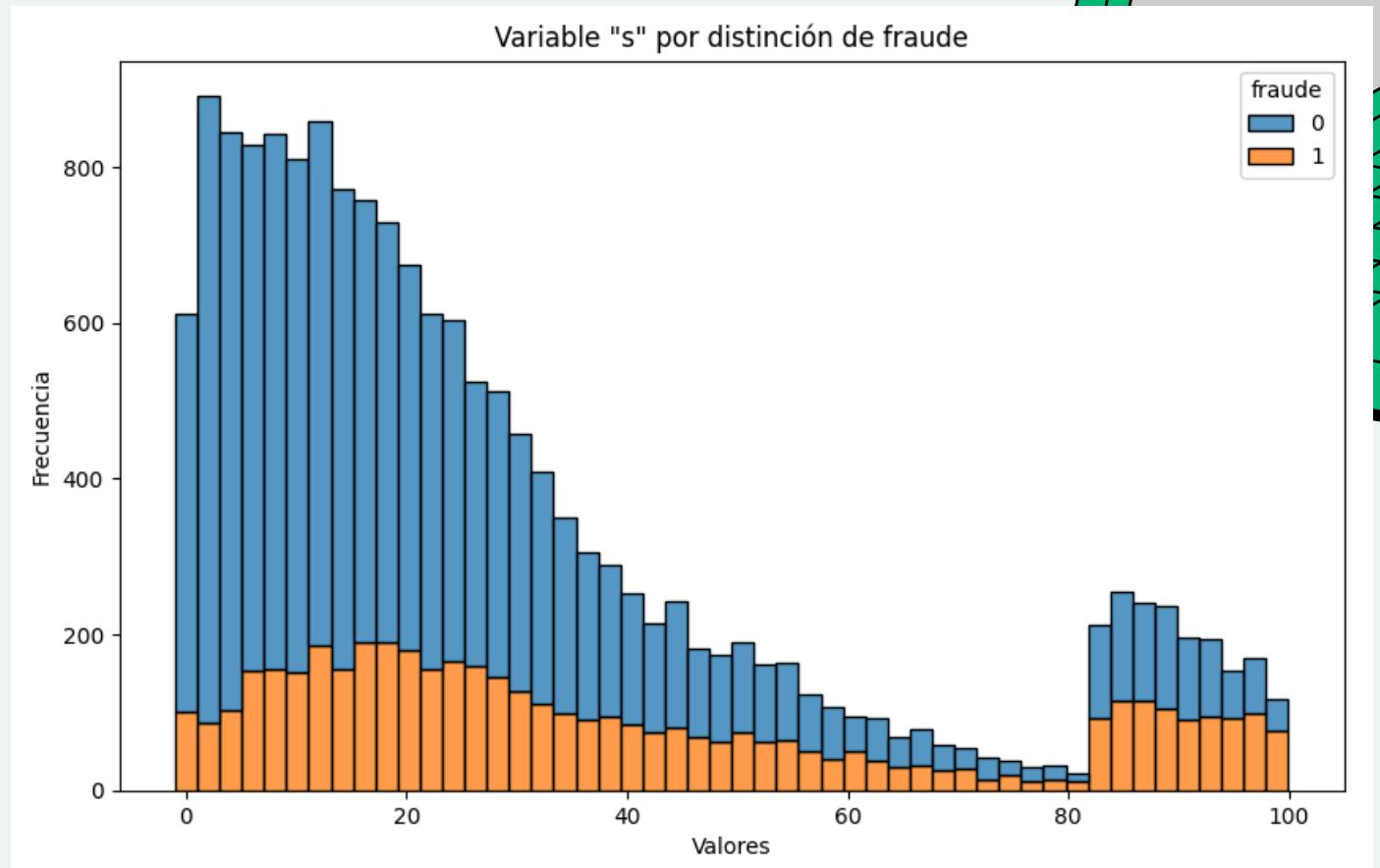
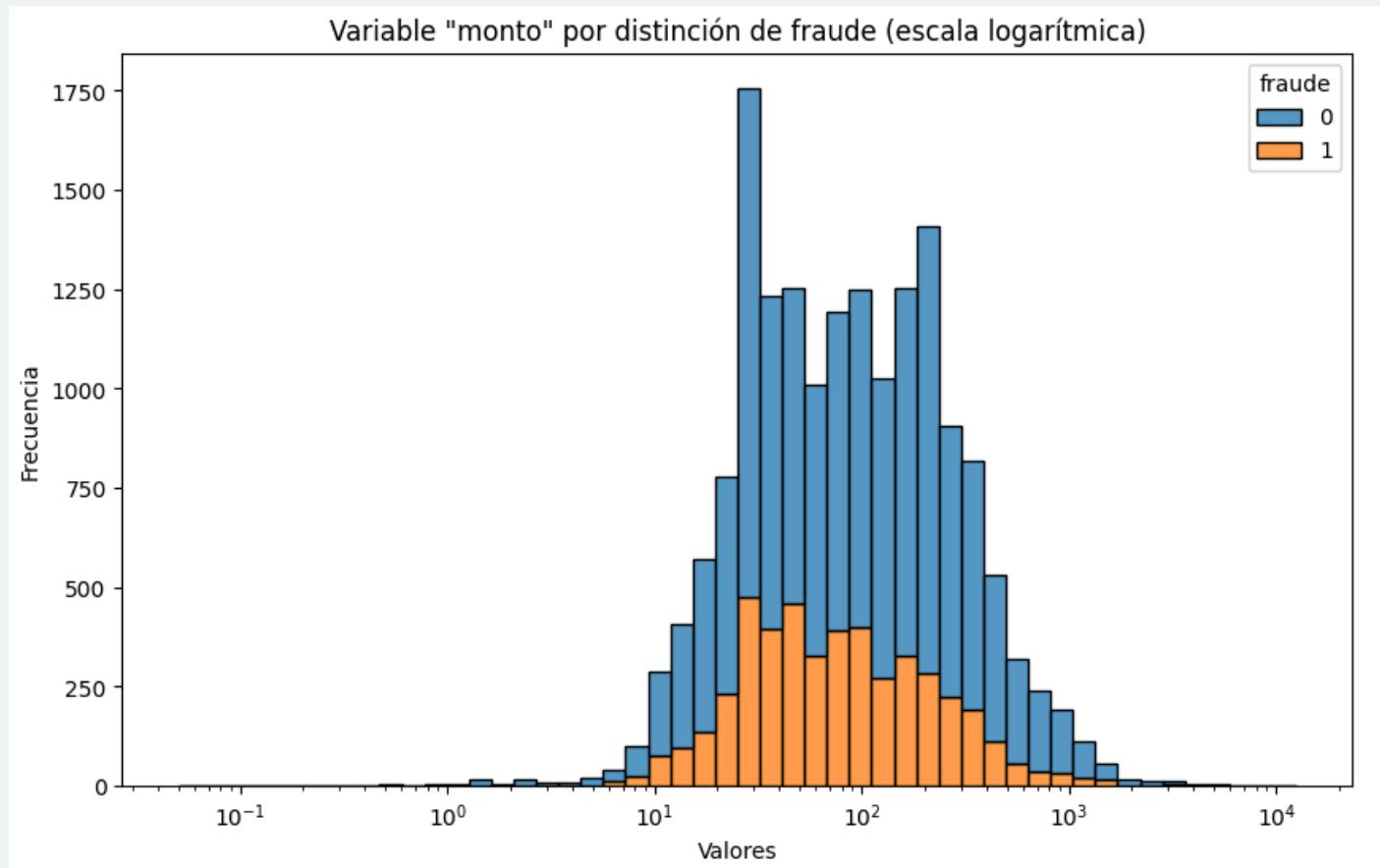
Análisis por países



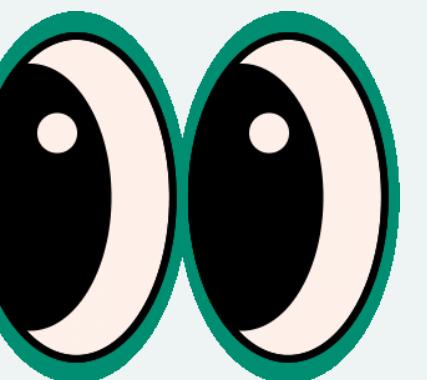
Distribuciones de variables numéricas



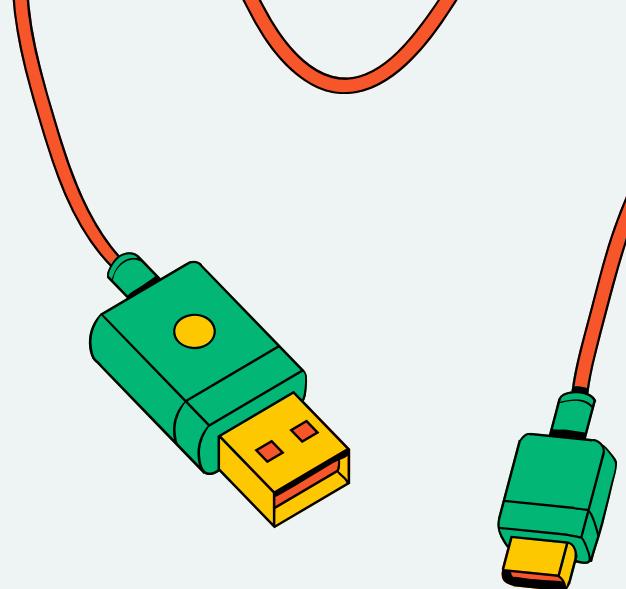
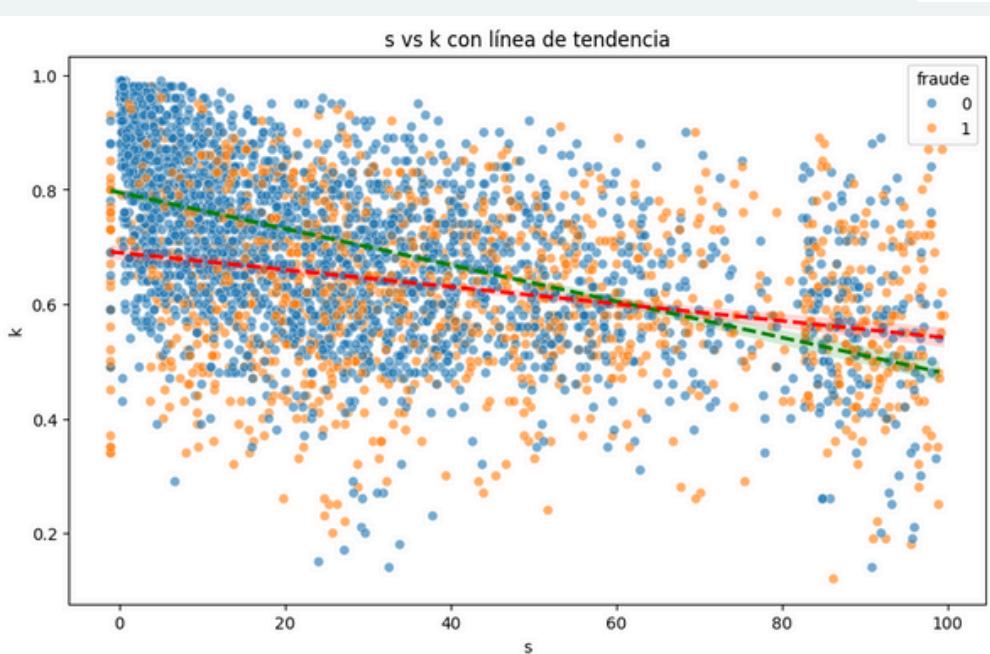
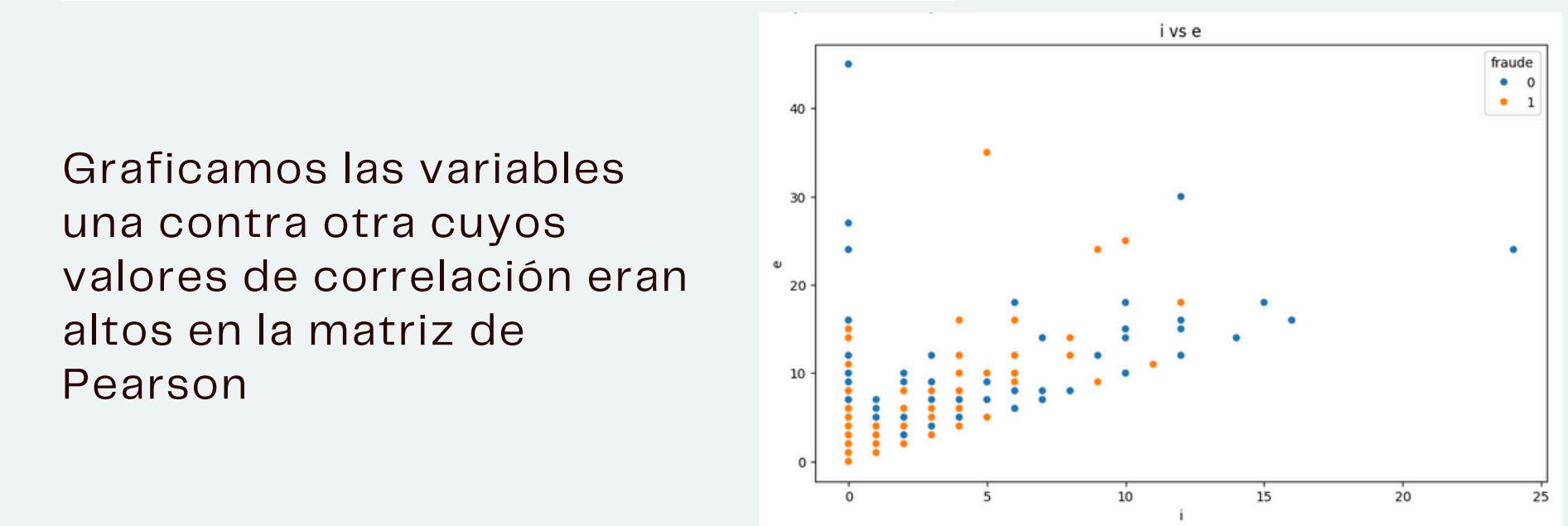
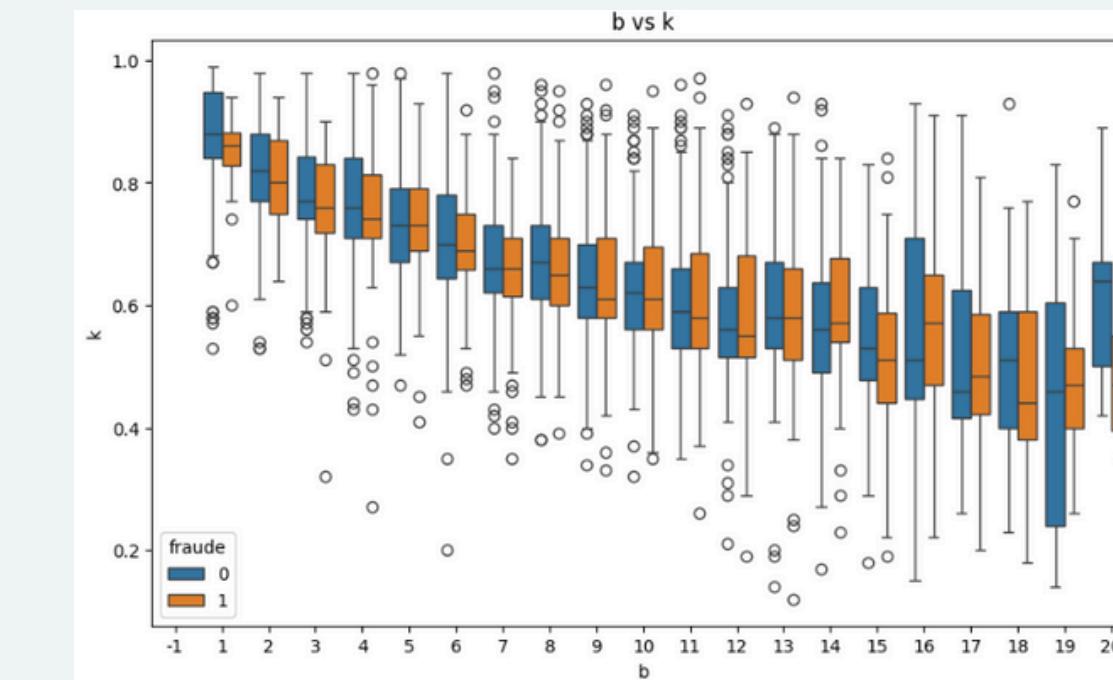
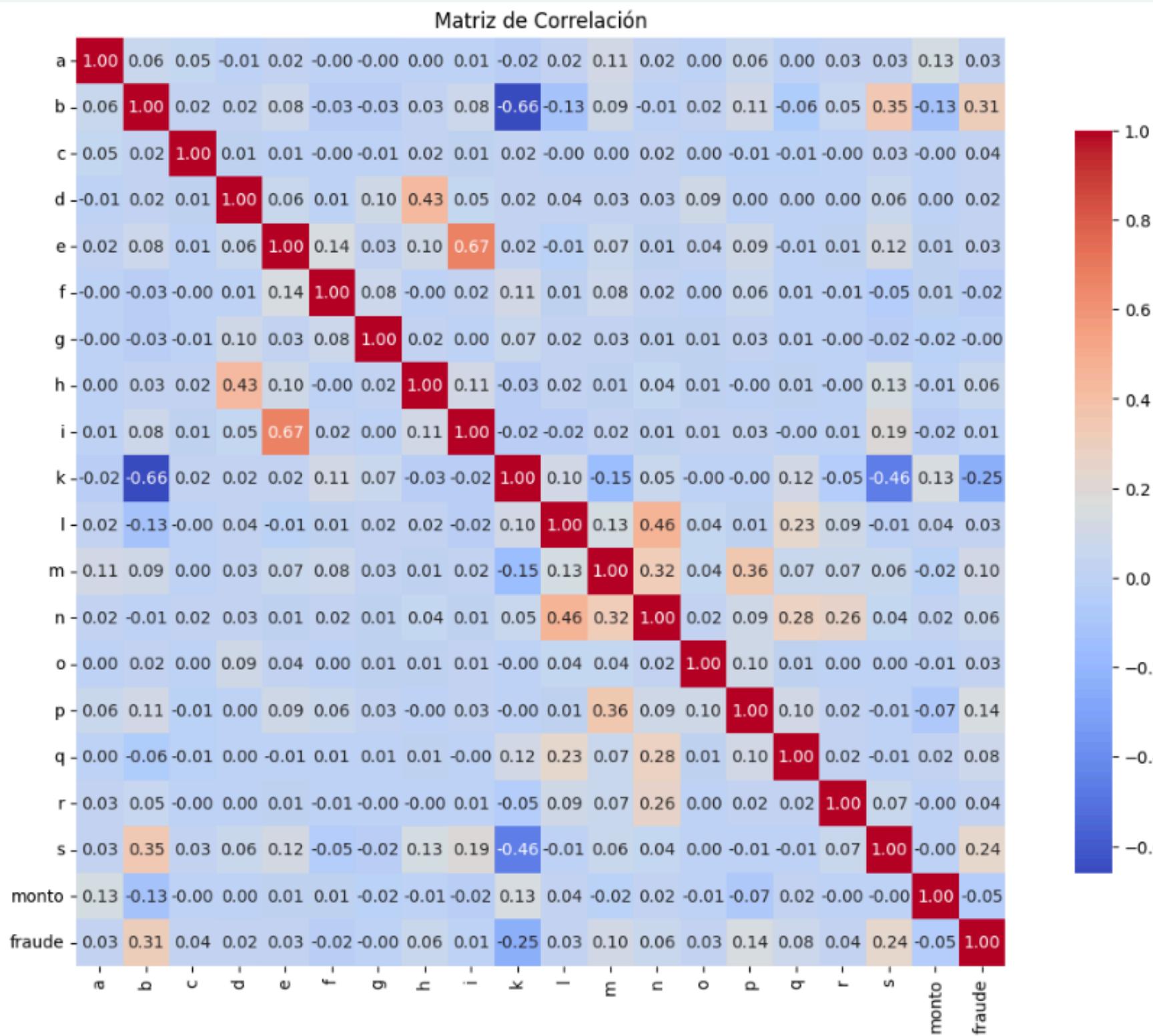
Distribuciones de variables numéricas



Estas variables y las mostradas anteriormente presentan un comportamiento particular: para ciertos valores de cada una, el fraude es mucho más notable que para otros, lo que podría indicarnos la importancia de cada una de ellas para elegir las features del modelo



Relación entre las variables



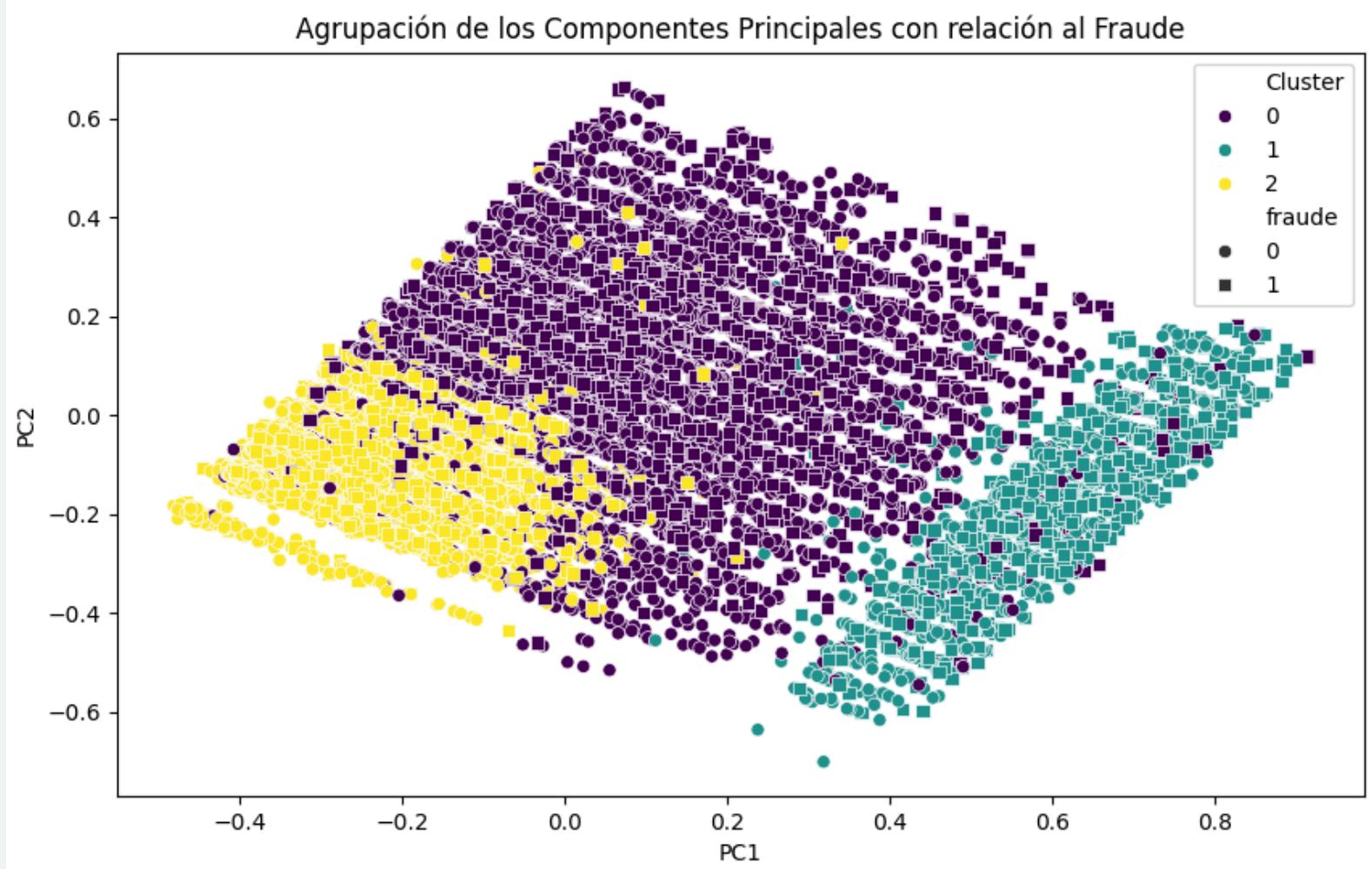
Análisis de Componentes Principales

PCA transforma las variables originales en componentes principales que capturan la mayor parte de la varianza del conjunto de datos lo que permite visualizar los datos en los primeros dos o tres componentes principales para observar patrones, tendencias y agrupaciones naturales.

Las 10 variables que más contribuyen al primer componente principal (PC1) son:

```
s    0.840989  
b    0.538272  
m    0.027081  
l    0.022713  
i    0.021950  
f    0.020281  
c    0.018001  
e    0.015837  
h    0.010264  
g    0.006198
```

Name: PC1, dtype: float64



¿QUÉ RELACIONES ENCONTRAMOS?



- A medida que el monto aumenta, parece haber una mayor proporción de transacciones fraudulentas. Esto podría indicar que es más probable encontrar fraude en montos más altos.
- Las gráficas sugieren que tanto los individuos que cometan fraude como los que no lo hacen exhiben patrones de comportamiento similares en relación a varias variables, lo que dificultará la detección del fraude.
- Para las variables **k**, **q**, **s**, **c** y **b**, las gráficas muestran que aunque el fraude ocurre en varios rangos de cada una, podría haber una ligera tendencia de fraude en ciertos intervalos. Encontrar pesos para esos intervalos es entonces la meta del modelo.



EMPEZANDO EL MODELO

01

DIVISIÓN DE LA DATA

Se toma el 20% de los datos para el conjunto de **testeo** y el 80% restante para el conjunto de **entrenamiento completo**. Se toma el 25% del conjunto de entrenamiento completo para el conjunto de validación y el 75% restante para el conjunto de entrenamiento final.

02

TRATAMIENTO DE NULOS

Hay solo dos features con datos nulos en el dataset. La primera es la columna **C donde llenaremos estos valores con la media de la misma variable** para cumplir con los supuestos del modelo. **Los valores nulos de la variable K serán llenados con cero**

03

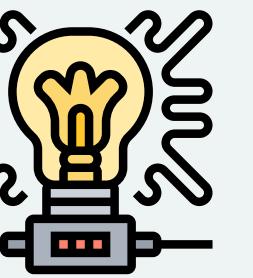
ESTANDARIZACIÓN DE LAS VARIABLES

Con el fin de evitar los outliers se normalizará toda la data numérica usando el **StandardScaler()** de scikit learn.

$$z = \frac{x - \mu}{\sigma}$$



FEATURE ENGINEERING



Se crearon dos nuevas columnas con la información de las features **J** y **K**.

Para K:

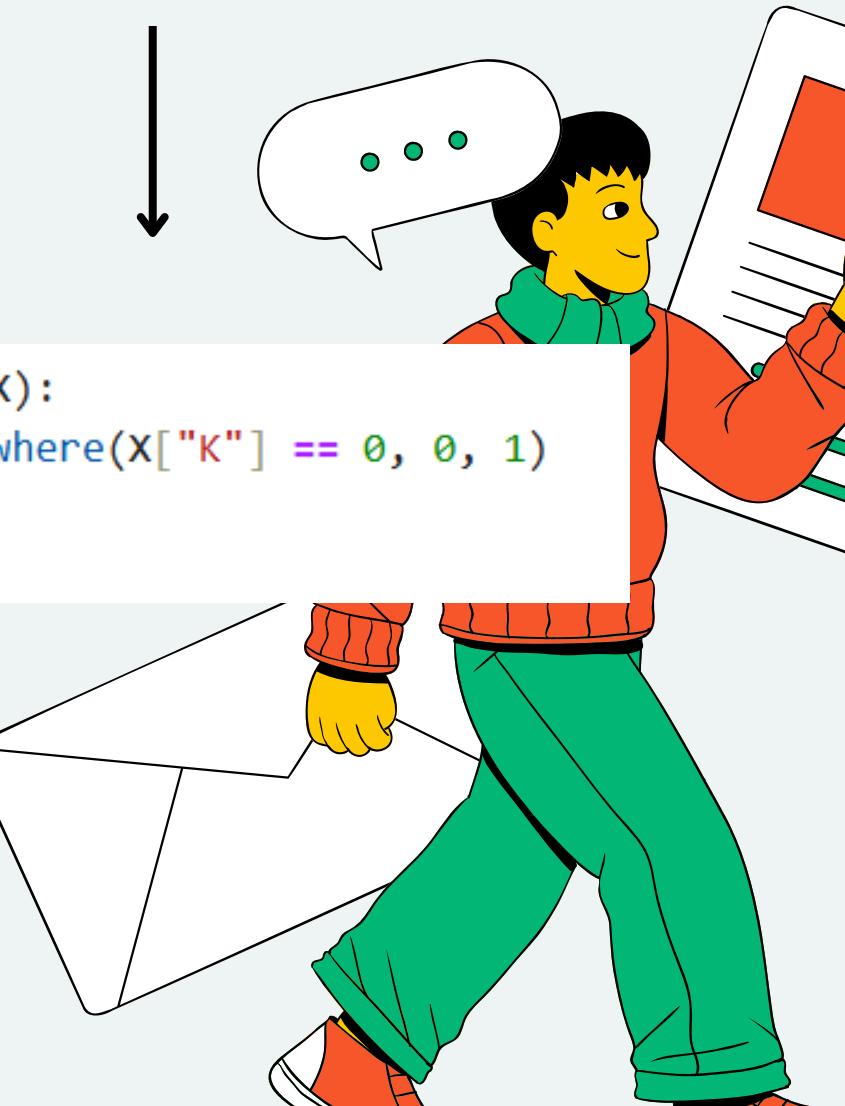
Como se observó en la gráfica de distribución de K, esta tenía una gran relación con respecto a lo que era fraude, sin embargo la mayoría de datos de esta variable eran nulos. Lo que se hizo fue crear una nueva columna llamada **k_bin** donde si K era nulo, ponía un **0** y si K tenía algún valor, ponía **1**

```
# creamos una nueva columna para cabin
df_train['k_bin'] = np.where(df_train['K'].isna(), 0, 1)
df_full_train['k_bin'] = np.where(df_full_train['K'].isna(), 0, 1)
df_test['k_bin'] = np.where(df_test['K'].isna(), 0, 1)
df_val['k_bin'] = np.where(df_val['K'].isna(), 0, 1)

# eliminamos cabin
del df_train['K']
del df_full_train['K']
del df_test['K']
del df_val['K']
```

Para el deploy del modelo este enfoque cambió un poco por cuestiones de facilidad: **teniendo en cuenta que ningún valor de K era cero, se llenaron los nulos con ceros** y ahora k_bin ponía cero donde K era 0 y 1 donde K era diferente de 0. La misma funcionalidad pero con un enfoque diferente

```
def transform(self, X):
    X["k_bin"] = np.where(X["K"] == 0, 0, 1)
    del X["K"]
    return X
```



Para J:

Para J se realizó un procedimiento más detallado. J tenía la información sobre el país de origen de la transacción. Como se observó en la gráfica habían países con una frecuencia más notable que otros. Siendo así se crearon tres grupos de la siguiente forma:

```
def assign_country_groups(df, country_counts):
    def assign_group(country):
        count = country_counts.get(country, 0)
        if count < 20:
            return 1
        elif 100 <= count < 350:
            return 2
        elif 1000 <= count <= 9400:
            return 3
        else:
            return 'otro'

    df['country_group'] = df['j'].apply(assign_group)
```

country_group	count
3	12907
2	572
1	25

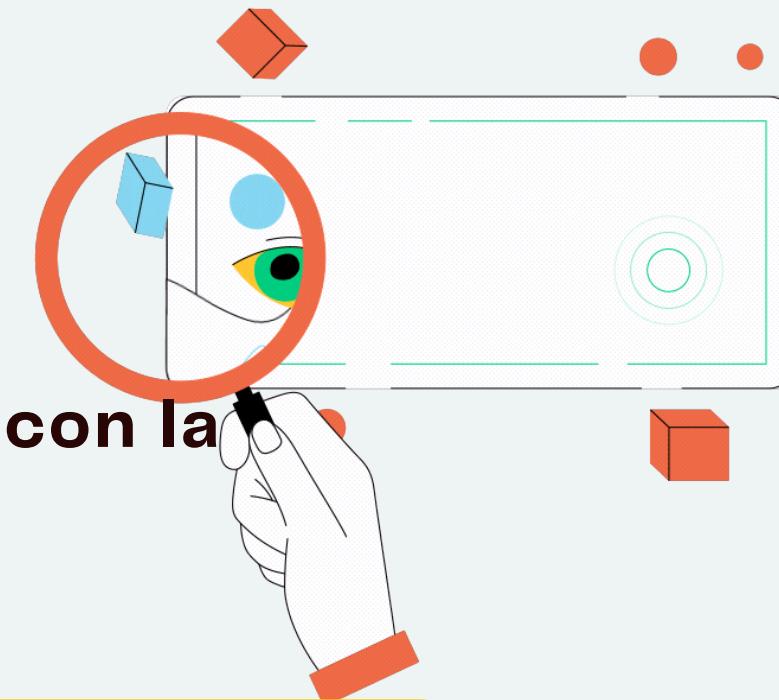
Se asignó un grupo diferente para los países que aparecieran entre 1 y 20 veces; entre 100 y 350 veces; y entre 1000 y 9400 veces



Luego se realizó One Hot Encoding a la columna country_group

FEATURES ESCOGIDAS

Con el fin de poder quedarnos con las features que mayor relación tuvieran con la detección de fraude, se llevaron a cabo varias metodologías



ARBOL DE DECISIÓN

Con la data ya numérica en su totalidad, se realiza un **DT** para ver cuáles son las features (según el índice de Gini) que mejor dividen el fraude y el no fraude

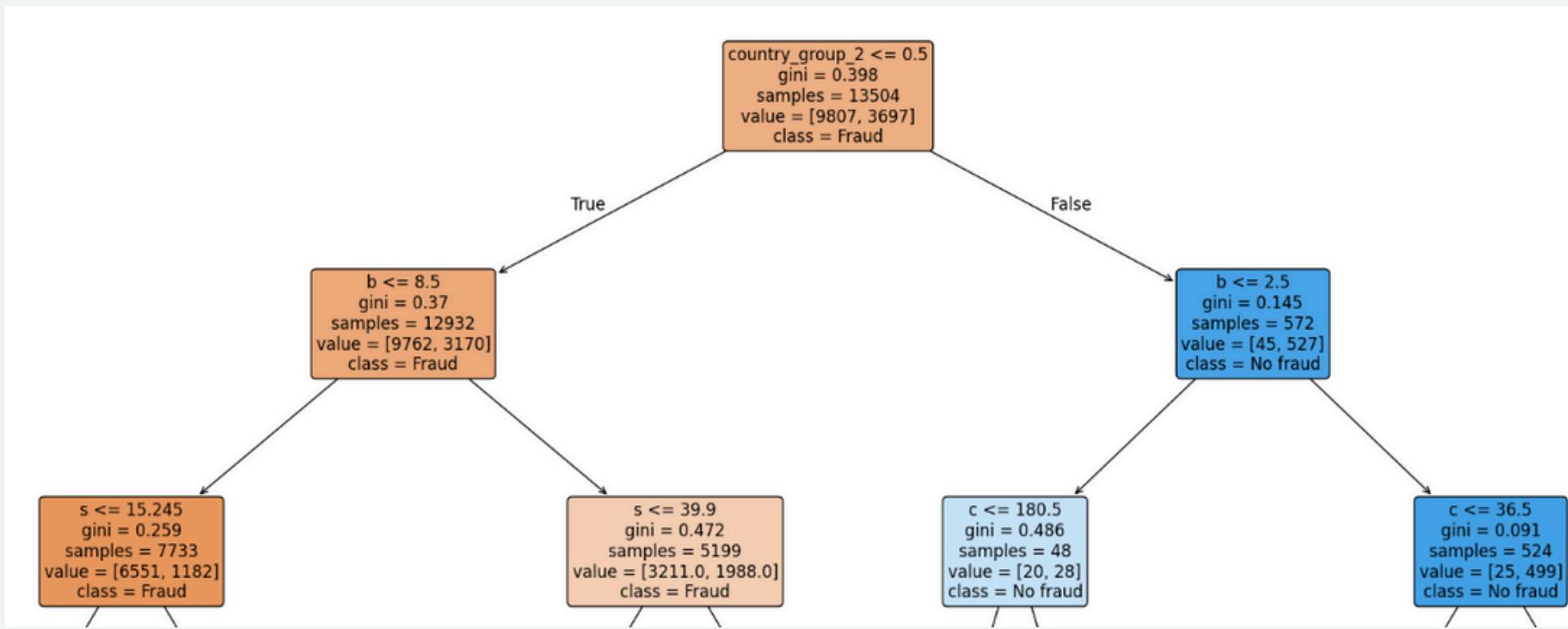
MUTUAL INFORMATION

Con la data numérica se calcula el **mutual_info_score** para ver cómo media del target, que es más una proporción, se relacionaba con la media de las demás features

PCA

PCA identifica las direcciones en las que los datos tienen mayor varianza. Las direcciones son combinaciones lineales de las **variables originales que mejor describen la dispersión de los datos alrededor de la media**.

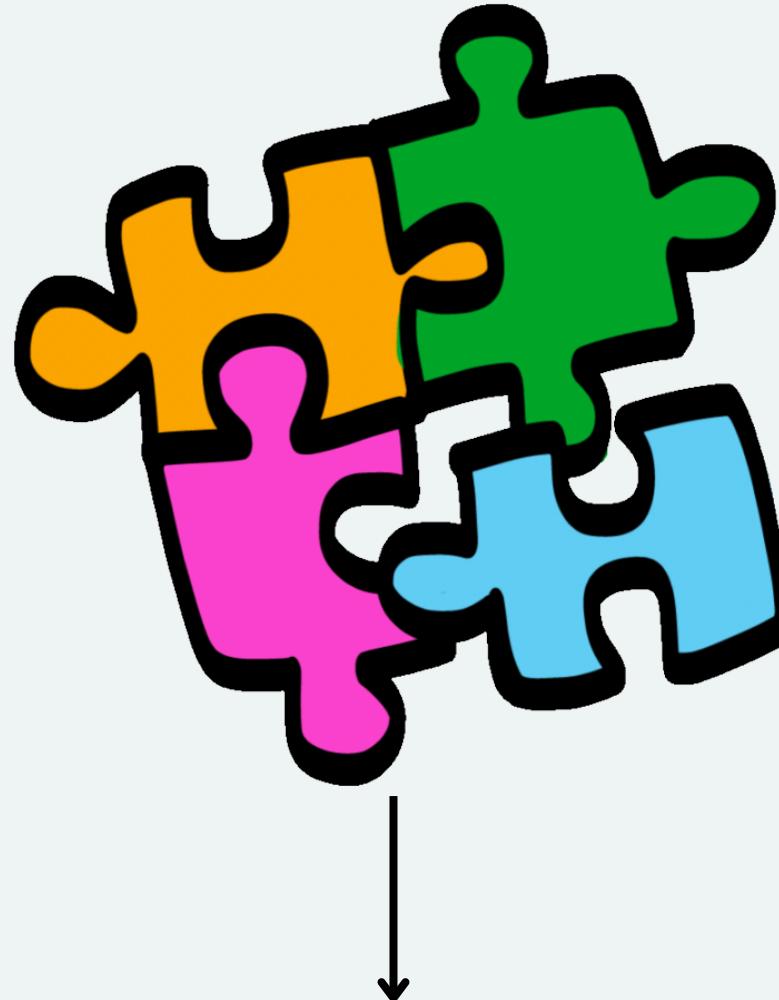




Features con la mayor ganancia de información para DT:
country_group_2, B, S, C, k_bin, M, N

	0
monto	0.451164
c	0.385711
s	0.309927
b	0.051461
country_group_2	0.041974
country_group_3	0.038187
q	0.028718
p	0.010875
f	0.007950

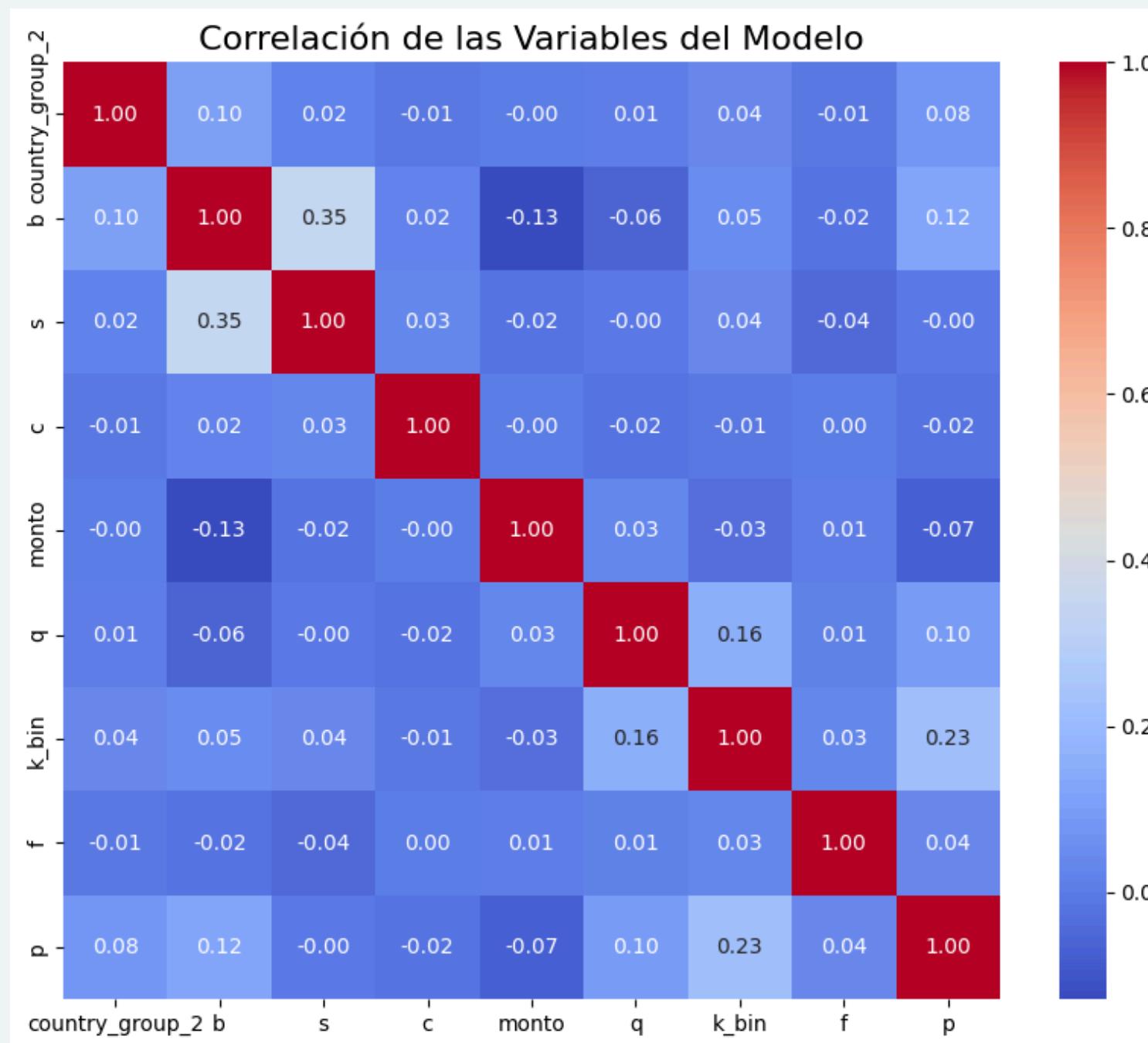
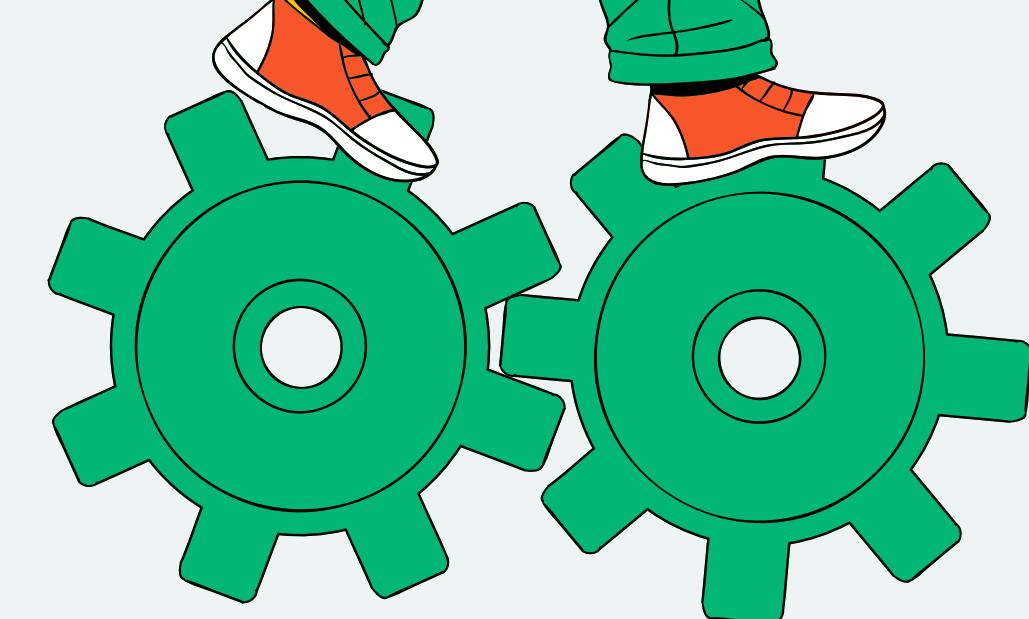
Features con la mayor relación con el target para mutual information:
Monto, C, B, S, country_group_2



Features con la mayor descripción de la varianza para PCA (mostradas anteriormente):
S, B, M, L, I, F, C

¡Encontramos entonces que todos ellos coinciden en varias variables!

FINALMENTE NOS QUEDAMOS CON:

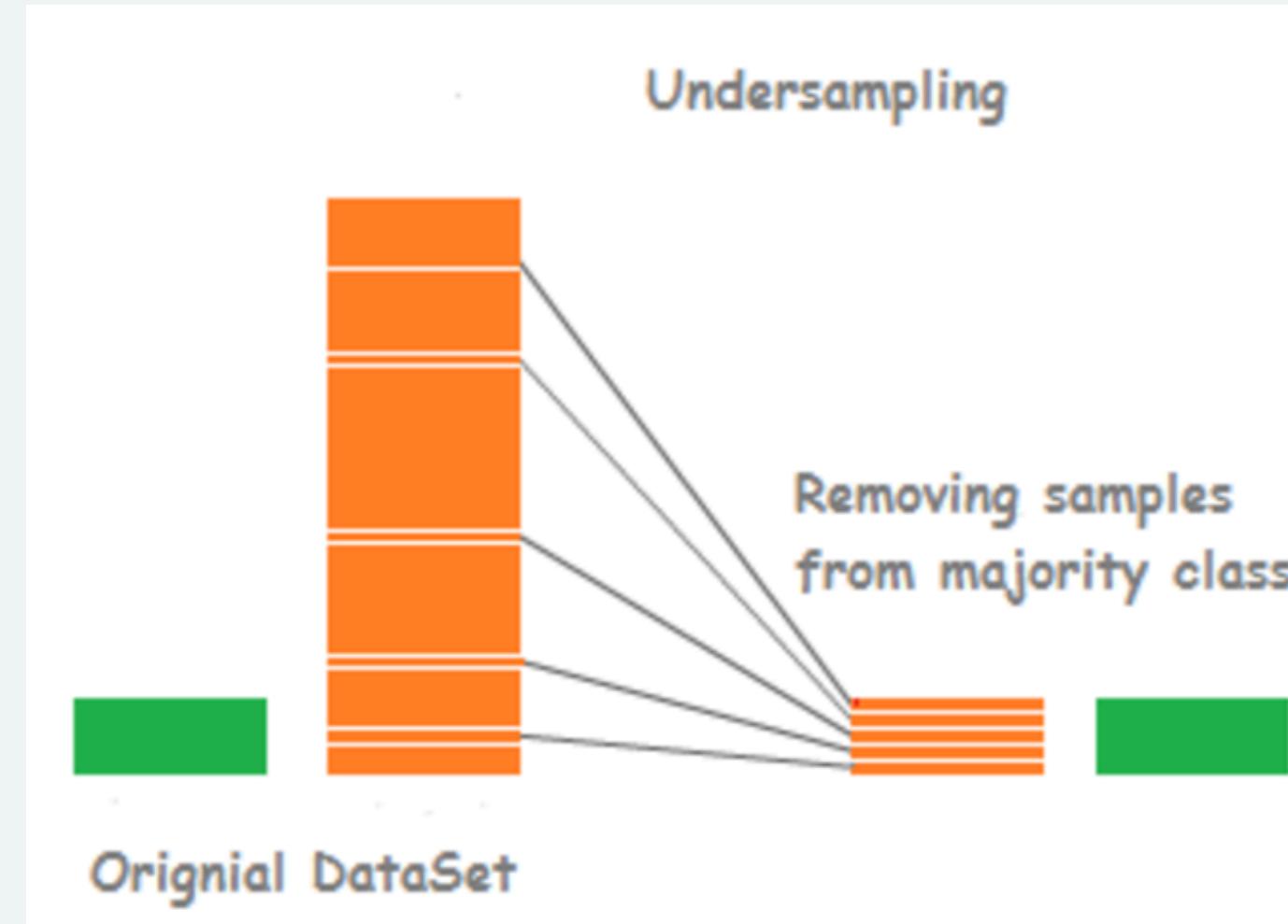


country_group_2, S, C, Monto, Q, K_bin, F, P, B, M, N

Recordando que las clases estaban desbalanceadas en aproximadamente un 70% no fraude 30% fraude, recurrimos a técnicas de **Resampling**

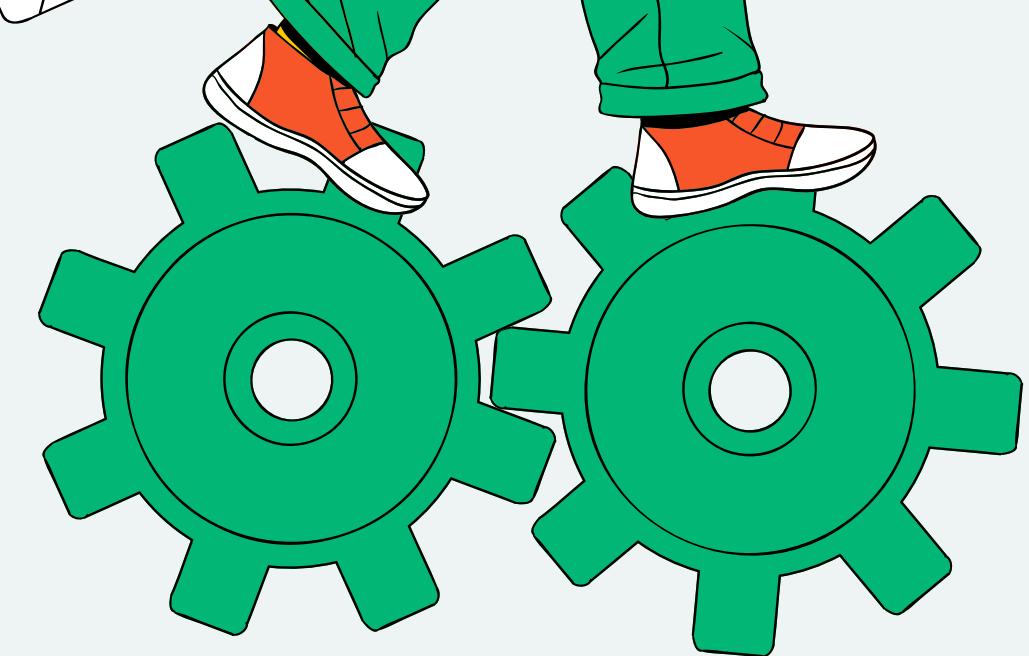


UNDERSAMPLING:

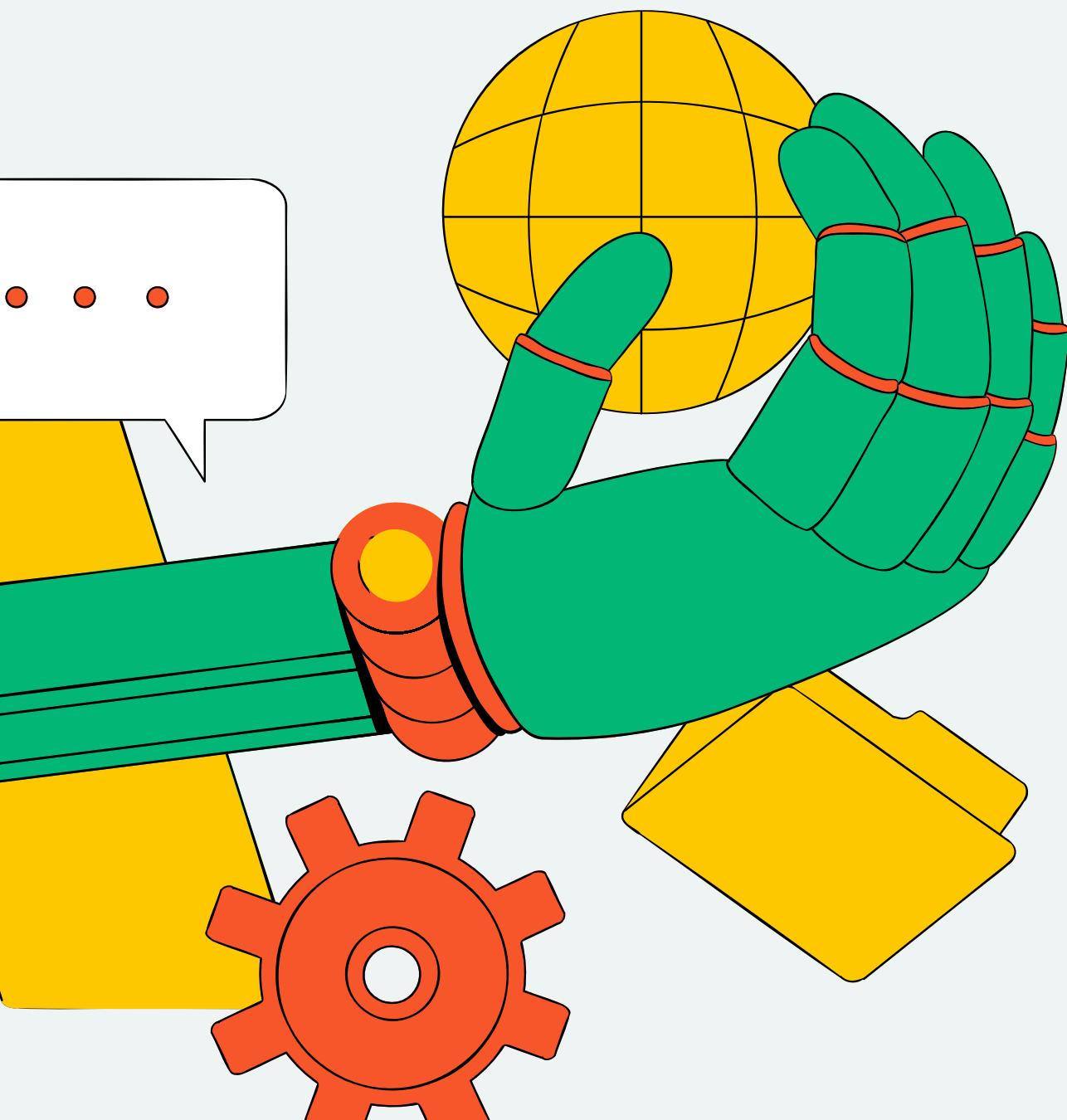


En esta técnica se reduce el tamaño de la clase mayoritaria para igualarlo o acercarlo al tamaño de la clase minoritaria. Esto se hace eliminando aleatoriamente muestras de la clase mayoritaria, y así ambas clases quedan aproximadamente balanceadas.

Para nuestro caso, tomamos que el número de ejemplos de la clase minoritaria fuera el 80% del número de ejemplos de la clase mayoritaria. Esto nos da un **total de 6259 datos** para el entrenamiento del modelo



MODELOS PROBADOS, MODELO ESCODIGO



Teniendo en cuenta que queremos detectar la mayor cantidad de fraude posible, nuestra prioridad es la métrica recall. Sin embargo en el contexto del problema, es importante también identificar lo que no es fraude: Mercado Libre no quisiera calificar como fraudulentas transacciones que no lo son porque esto se traduciría en pérdidas monetarias.

Es por eso entonces que se **priorizó el recall y el f1-score**.



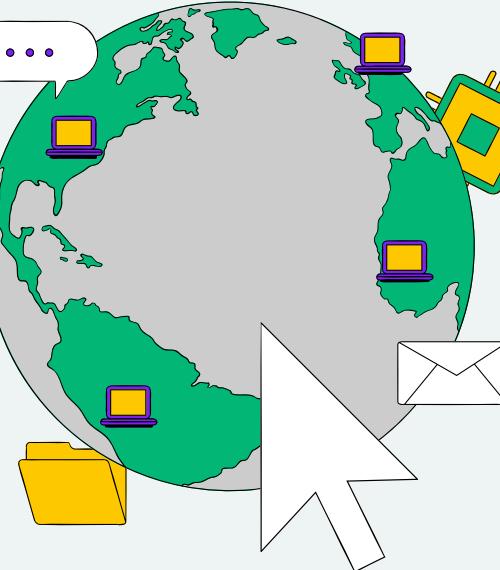
PRUEBA DE MODELOS: AB, XGB, DT, RF, GB



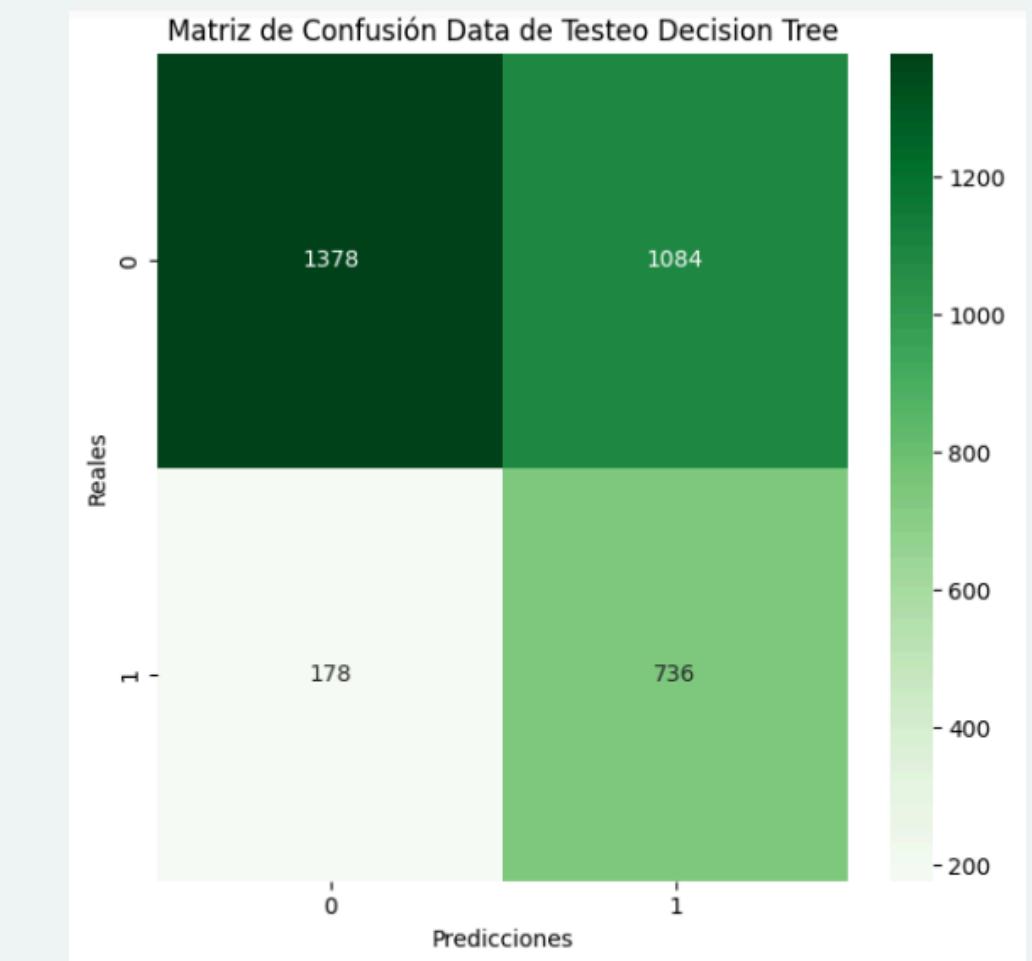
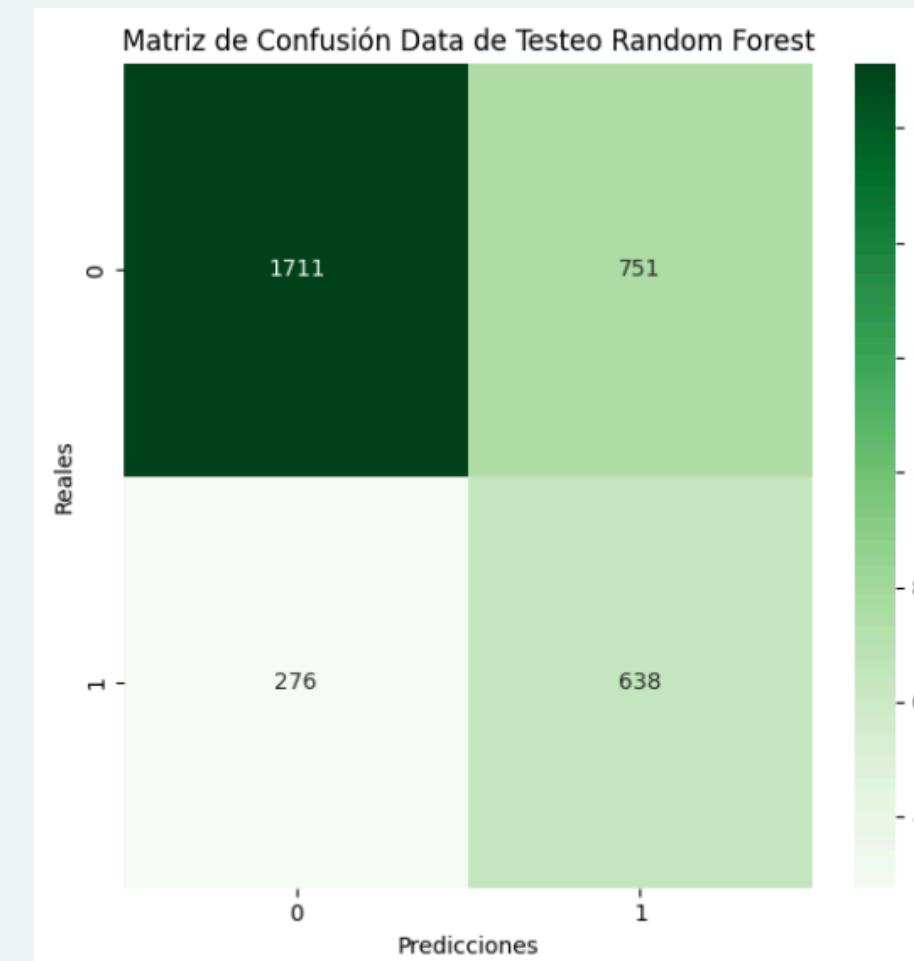
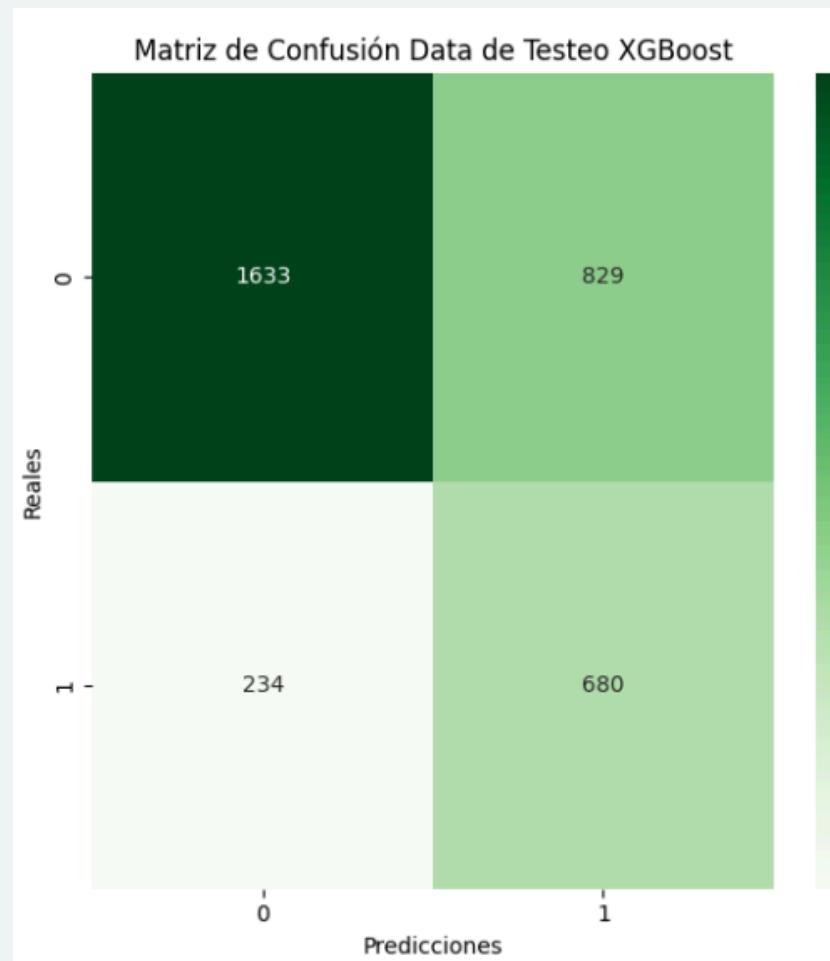
AdaBoost, XGBoost, Decision Tree, Random Forest y Gradient Boost fueron probados para mirar el performance del modelo, LR no fue usado ya que dada la robustez del problema, el algoritmo no era lo suficientemente bueno con los patrones de las variables

Modelo	Testeo	Validación
RF	0.70-0.55	0.68-0.54
GB	0.54-0.54	0.60-0.55
AB	0.47-0.52	0.52-0.54
XGB	0.74-0.56	0.78-0.56
DT	0.81-0.54	0.79-0.53





XGBOOST, RANDOM FOREST, DECISION TREE

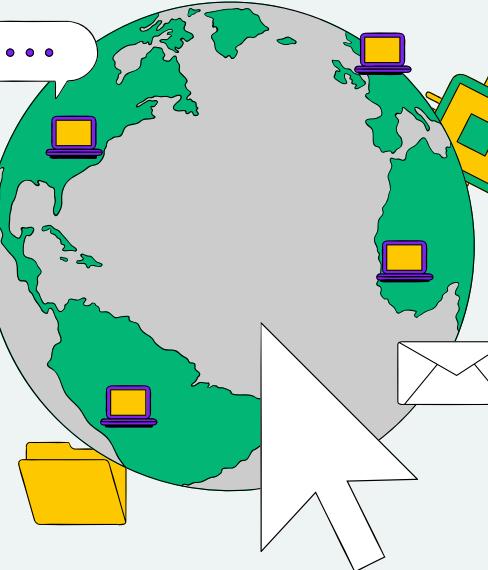


Accuracy: 0.69
Precision: 0.45
Recall: 0.74
F1 Score: 0.56

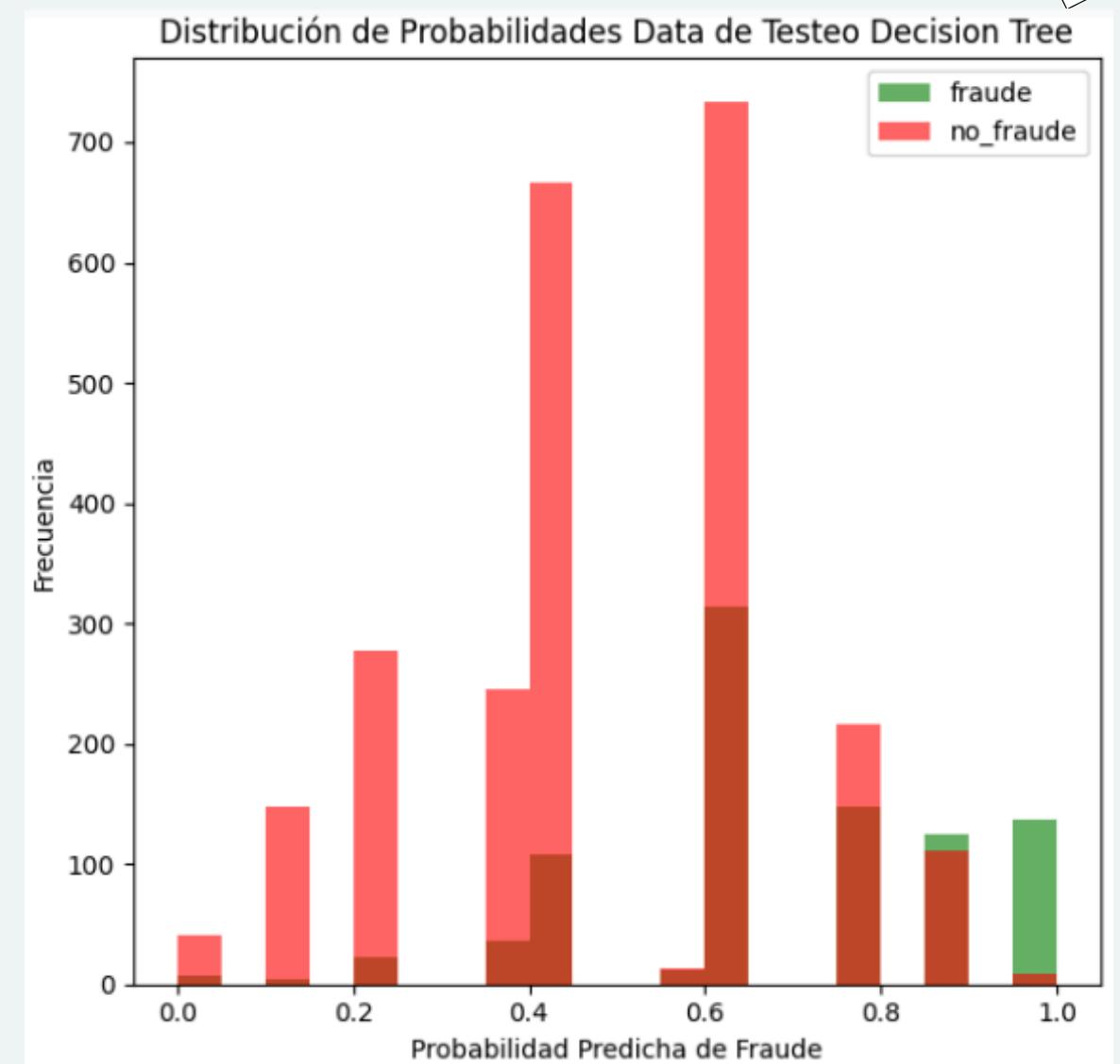
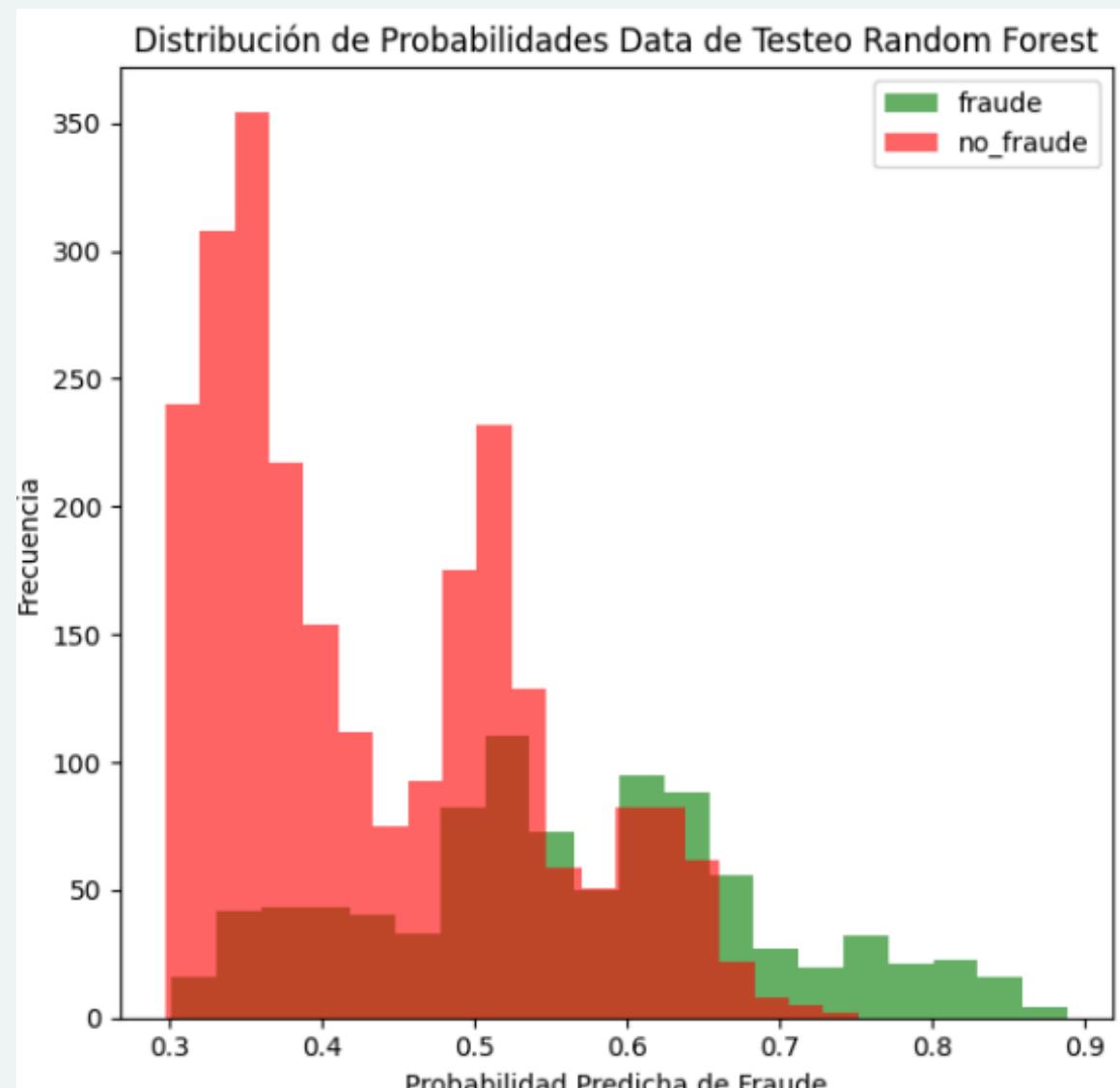
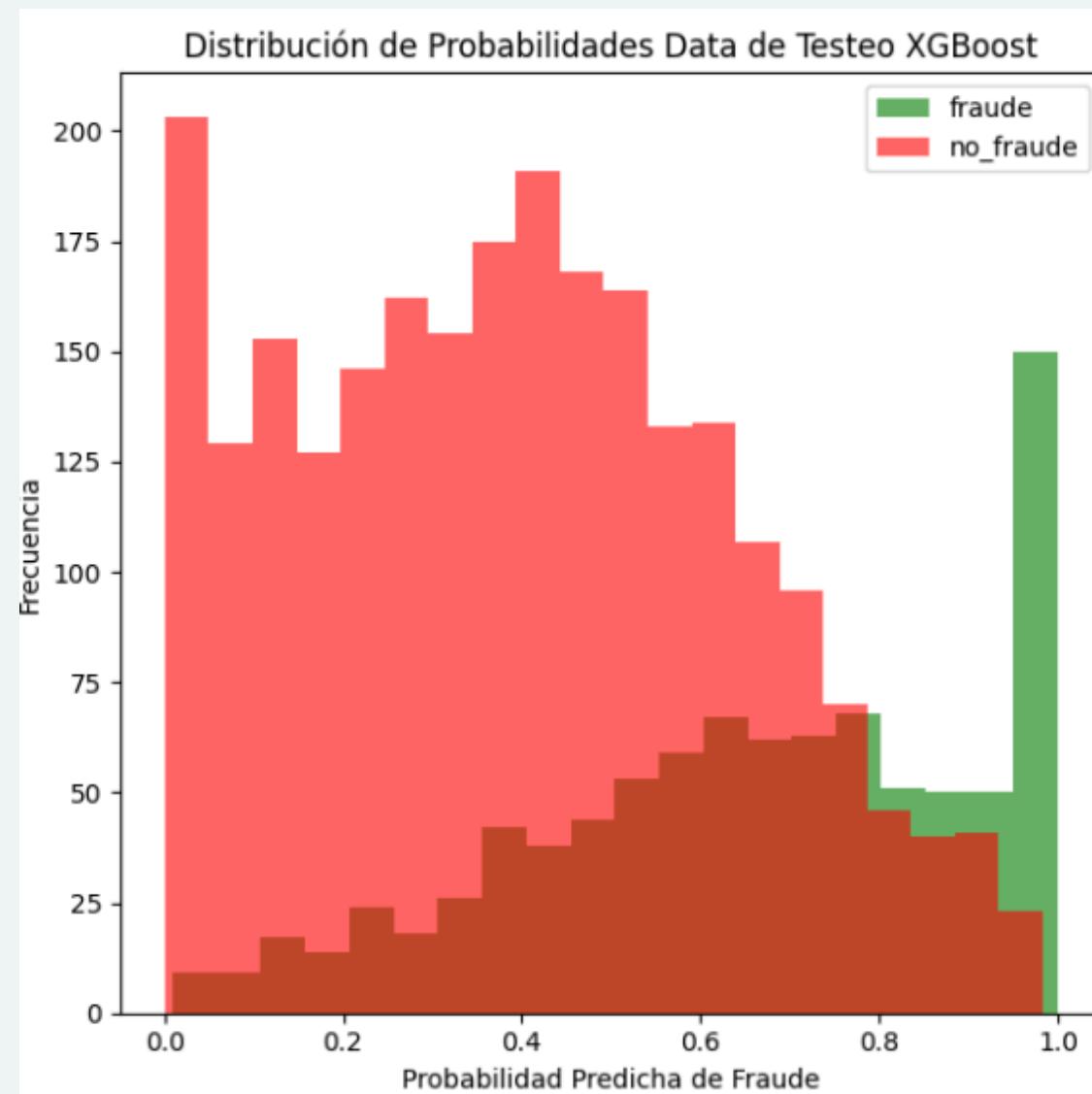
Accuracy: 0.70
Precision: 0.46
Recall: 0.70
F1 Score: 0.55

Accuracy: 0.63
Precision: 0.40
Recall: 0.81
F1 Score: 0.54





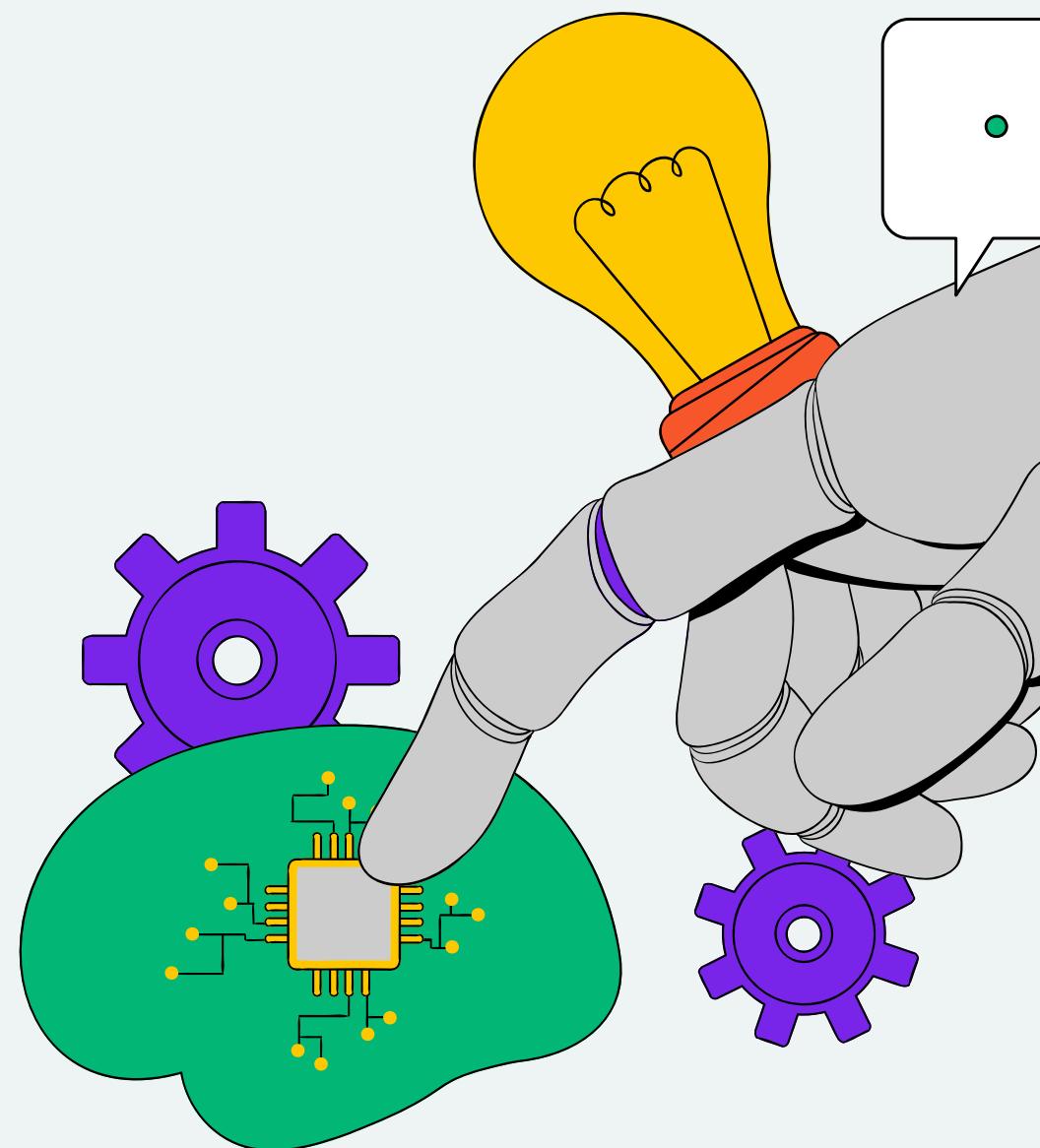
XGBOOST, RANDOM FOREST, DECISION TREE



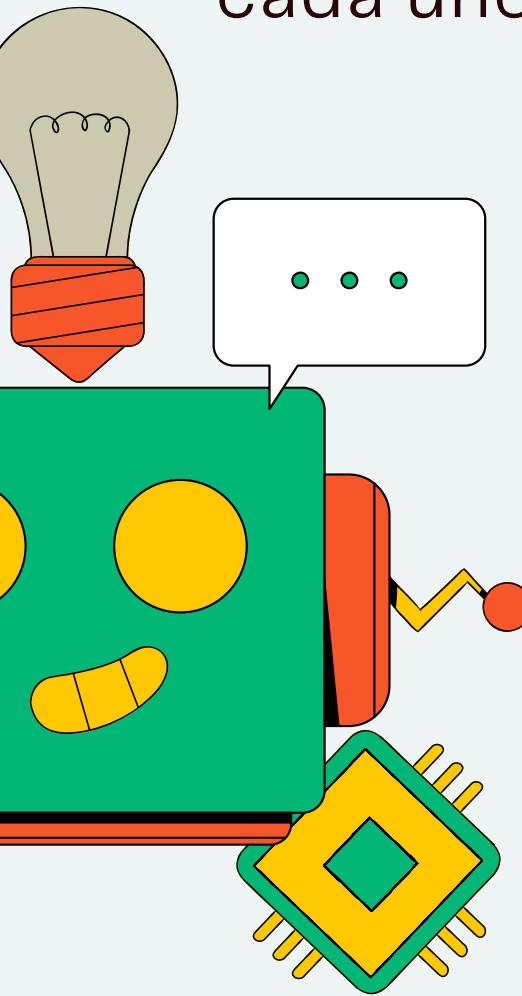
INSIGHTS:

- El modelo con XGB logra un mejor equilibrio entre la detección de fraudes reales y una reducción de los falsos negativos, sin embargo, el modelo aún clasifica un número considerable de transacciones legítimas como fraude.
- El modelo con RF muestra una separación notable en el histograma de probabilidades. Sin embargo en la matriz de confusión podemos ver que en la búsqueda de equilibrio entre falsos positivos y positivos sacrifica muchos casos de fraude.
- El modelo de DT tiene una alta precisión al identificar las transacciones que no son fraudulentas, pero está dejando de lado un poco las transacciones fraudulentas.

¿Qué modelo elegir entonces?



Para escoger el modelo que usaremos para el despliegue del mismo usaremos como criterio la matriz de confusión resultante de cada uno de los algoritmos mostrados



$$CM_{XGB} = \begin{pmatrix} 1633 & 829 \\ 234 & 680 \end{pmatrix}$$

$$CM_{RF} = \begin{pmatrix} 1711 & 751 \\ 276 & 638 \end{pmatrix}$$

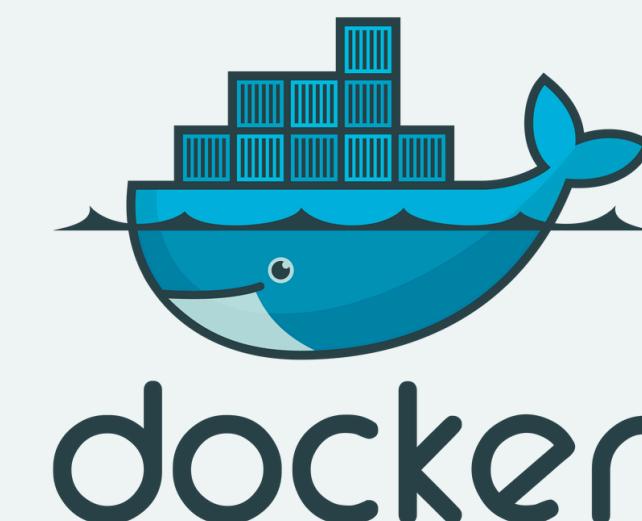
eo):

$$CM_{DT} = \begin{pmatrix} 1378 & 1084 \\ 178 & 736 \end{pmatrix}$$

A pesar del inconveniente que se mencionó antes para XGBoost, este muestra un rendimiento superior al priorizar el recall, lo que resulta crucial para el contexto de detección de fraudes.



Así pues, la decisión final para el despliegue del modelo con Docker y FastAPI se hará con XGBoost, debido a que después de realizar la evaluación de varios hiperparámetros, la matriz de confusión nos da mejores resultados que los otros dos modelos.



DEPLOY DEL MODELO:



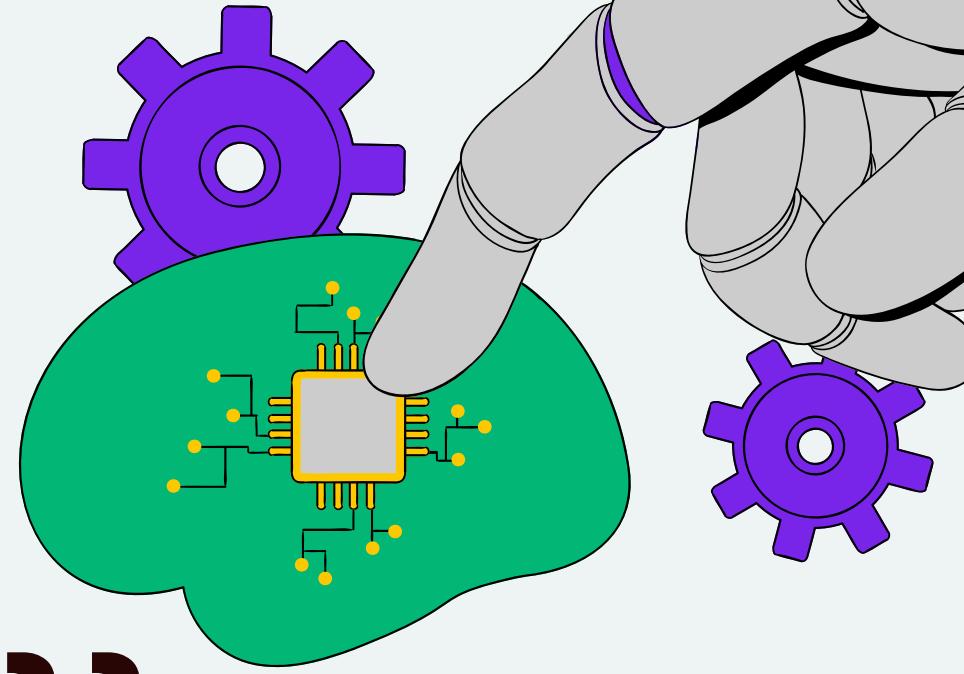
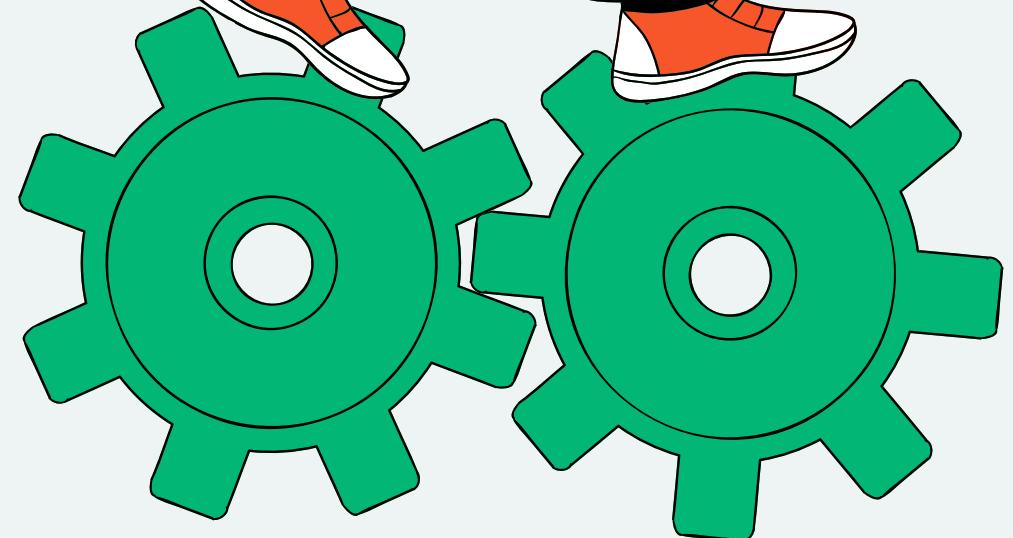
```
11      "J": "FG",
12      "K": 0.58,
13      "L": 0,
14      "M": 3,
15      "N": 1,
16      "O": 5,
17      "P": 1,
18      "Q": 1,
19      "R": 1,
20      "S": 444,
21      "Monto": 12
22  }

Body  ⚡  200 OK • 69 ms
Pretty Raw Preview Visualize JSON ↗
1  {
2    "Fraud": "papi nos robaron"
3 }
```



Para el deploy del modelo y su contexto, leer por favor los comentarios del notebook “FinalExam_Deploy.ipynb”





iii GRACIAAAS!!! :)]

