```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('/train.csv')
data.head(5)
```

	id	FRAUDE	VALOR	HORA_AUX	Dist_max_NAL	Canal1	FECHA	COD_PAIS	CANAL	DIAS
0	9000000001	1	0.0	13	659.13	ATM_INT	20150501	US	ATM_INT	
1	9000000002	1	0.0	17	594.77	ATM_INT	20150515	US	ATM_INT	
2	9000000003	1	0.0	13	659.13	ATM_INT	20150501	US	ATM_INT	
3	9000000004	1	0.0	13	659.13	ATM_INT	20150501	US	ATM_INT	
4	9000000005	1	0.0	0	1.00	ATM_INT	20150510	CR	ATM_INT	

5 rows × 26 columns

```
data.describe()
```

	id	FRAUDE	VALOR	HORA_AUX	Dist_max_NAL	FECHA	DIASE
count	2.965000e+03	2965.000000	2.965000e+03	2965.000000	2965.000000	2.965000e+03	2965.00000
mean	6.890938e+09	0.246543	5.035695e+05	14.960877	314.656739	2.015051e+07	3.14300
std	9.739700e+09	0.431071	9.859497e+05	6.348607	295.142673	9.134641e+00	2.09228
min	2.364560e+06	0.000000	0.000000e+00	0.000000	1.000000	2.015050e+07	0.00000
25%	2.552997e+09	0.000000	9.016001e+04	12.000000	24.830000	2.015050e+07	1.00000
50%	6.142884e+09	0.000000	2.435912e+05	16.000000	243.620000	2.015052e+07	3.00000
75%	9.000000e+09	0.000000	5.058190e+05	20.000000	594.770000	2.015052e+07	5.00000
max	9.330050e+10	1.000000	2.001406e+07	23.000000	1310.460000	2.015053e+07	6.00000

8 rows × 21 columns

```
number_fraude = len(data[data.FRAUDE == 1])
number_normal = len(data[data.FRAUDE == 0])

print ("Fraude:", number_fraude)
print ("Normal:", number_normal)
```

```
Fraude: 731
Normal: 2234
```

DATA CLEANING

```
# Identificar valores nulos
print(data.isnull().sum())
```

```
id                   0
FRAUDE               0
VALOR                0
HORA_AUX             0
Dist_max_NAL         0
Canal1               0
FECHA                0
COD_PAIS             0
CANAL                0
DIASEM               0
DIAMES               0
FECHA_VIN           24
OFICINA_VIN         24
SEXO                55
SEGMENTO            24
EDAD                24
INGRESOS            24
EGRESOS             24
NROPAISES            0
Dist_Sum_INTER    1547
Dist_Mean_INTER   1547
Dist_Max_INTER    1547
NROCIUDADES          0
Dist_Mean_NAL      457
Dist_HOY             0
```

```
Dist_sum_NAL          0
dtype: int64
```

```
# Hipotesis: si numero de paises es igual a '1' las medidas internacionales aparecen como 'NAN', por tanto es factible remplezar esos

# Filtrar las filas donde NROPAISES es igual a 1
subset_data = data[data['NROPAISES'] == 1]

# Verificar si las columnas son nulas en el subconjunto
print("Número de filas donde NROPAISES es 1:", len(subset_data))

# Comprobar si las columnas son nulas
print("Número de filas donde Dist_Sum_INTER es nulo:", subset_data['Dist_Sum_INTER'].isnull().sum())
print("Número de filas donde Dist_Mean_INTER es nulo:", subset_data['Dist_Mean_INTER'].isnull().sum())
print("Número de filas donde Dist_Max_INTER es nulo:", subset_data['Dist_Max_INTER'].isnull().sum())
```

```
Número de filas donde NROPAISES es 1: 1547
Número de filas donde Dist_Sum_INTER es nulo: 1547
Número de filas donde Dist_Mean_INTER es nulo: 1547
Número de filas donde Dist_Max_INTER es nulo: 1547
```

```
# Seria valido reemplzar los NAN por 0
data['Dist_Sum_INTER'].fillna(0, inplace=True)
data['Dist_Mean_INTER'].fillna(0, inplace=True)
data['Dist_Max_INTER'].fillna(0, inplace=True)
```

```
# Filtrar las filas con valores nulos en las columnas mencionadas
filas_nulas = data.loc[data['FECHA_VIN'].isnull() | data['OFICINA_VIN'].isnull() | data['SEGMENTO'].isnull() | data['EDAD'].isnull() | data['INGRESOS'].isn

# Mostrar las filas que cumplen con la condición
filas_nulas.isnull().sum()

#La hipotesis parece correcta, y tales datos podrian corresponder a datos de transacciones fallidas
```

```
id                  0
FRAUDE              0
VALOR               0
HORA_AUX            0
Dist_max_NAL        0
Canal1              0
FECHA               0
COD_PAIS            0
CANAL               0
DIASEM              0
DIAMES              0
FECHA_VIN          24
OFICINA_VIN        24
SEXO               24
SEGMENTO           24
EDAD               24
INGRESOS           24
EGRESOS            24
NROPAISES           0
Dist_Sum_INTER      0
Dist_Mean_INTER     0
Dist_Max_INTER      0
NROCIUDADES         0
Dist_Mean_NAL       8
Dist_HOY            0
Dist_sum_NAL        0
dtype: int64
```

```
transacciones_normales_nulos = filas_nulas[filas_nulas['FRAUDE'] == 0].shape[0]

print('Transacciones normales entre las filas nulas: ', transacciones_normales_nulos)
```

```
Transacciones normales entre las filas nulas:  23
```

```
# Eliminar filas nulas de 'filas_nulas' en el DataFrame original 'data'
data_filter = data.drop(filas_nulas.index)
```

```
data=data_filter
# suficiente para descartar todas estas columnas nulas incluyendo la que aparece como fraude, por tanto descartamos estas columnas
```

```
# Hipotesis: si el NROCIUDADES (numero de ciudades) es 1, Dist_Mean_NAL es NAN, si la hiposis es correcta se puede cambiar los valores NAN como '0'


# Filtrar las filas donde NROCIUDADES es igual a 1
subset_data = data[data['NROCIUDADES'] == 1]

# Verificar si las columnas son nulas en el subconjunto
print("Número de filas donde NROCIUDADES es 1:", len(subset_data))

# Comprobar si las columnas son nulas
print("Número de filas donde Dist_Mean_NAL es nulo:", subset_data['Dist_Mean_NAL'].isnull().sum())
```

```
Número de filas donde NROCIUDADES es 1: 449
Número de filas donde Dist_Mean_NAL es nulo: 449
```

```
#La hipotesis es correcta, cambiamos '0' por 'NAN'

data['Dist_Mean_NAL'].fillna(0, inplace=True)
```

Por ultimo faltaria hacer un tratamiento al los datos que no reportan sexo, no obstante al ser datos categoricos ya no binario, y ser relativamente pocos podriamos implementar cualquier metodo, sin embargo para efectors practicos y teniendo en cuentas que son relativamente pocos datos simplemente los borrare.

```
filas_nulas_sexo = data[data['SEXO'].isnull()]

# Contar la cantidad de fraudes en este subconjunto
fraudes_en_filas_nulas = filas_nulas_sexo['FRAUDE'].sum()
print('Número de fraudes en filas con sexo nulo:', fraudes_en_filas_nulas)
```

```
Número de fraudes en filas con sexo nulo: 8
```

```
data = data.dropna(subset=['SEXO'])
```

```
print(data.isnull().sum())
```

```
id                0
FRAUDE            0
VALOR             0
HORA_AUX          0
Dist_max_NAL      0
Canal1            0
FECHA             0
COD_PAIS          0
CANAL             0
DIASEM            0
DIAMES            0
FECHA_VIN         0
OFICINA_VIN       0
SEXO              0
SEGMENTO          0
EDAD              0
INGRESOS          0
EGRESOS           0
NROPAISES         0
Dist_Sum_INTER    0
Dist_Mean_INTER   0
Dist_Max_INTER    0
NROCIUDADES       0
Dist_Mean_NAL     0
Dist_HOY          0
Dist_sum_NAL      0
dtype: int64
```

```
# Suponiendo que 'data_sin_nulas' es tu DataFrame sin filas nulas
data.to_csv('/data_preliminar.csv', index=False)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2910 entries, 0 to 2964
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               2910 non-null   int64
```

```
 1   FRAUDE           2910 non-null   int64
 2   VALOR            2910 non-null   float64
 3   HORA_AUX         2910 non-null   int64
 4   Dist_max_NAL     2910 non-null   float64
 5   Canal1           2910 non-null   object
 6   FECHA            2910 non-null   int64
 7   COD_PAIS         2910 non-null   object
 8   CANAL            2910 non-null   object
 9   DIASEM           2910 non-null   int64
 10  DIAMES           2910 non-null   int64
 11  FECHA_VIN        2910 non-null   float64
 12  OFICINA_VIN      2910 non-null   float64
 13  SEXO             2910 non-null   object
 14  SEGMENTO         2910 non-null   object
 15  EDAD             2910 non-null   float64
 16  INGRESOS         2910 non-null   float64
 17  EGRESOS          2910 non-null   float64
 18  NROPAISES        2910 non-null   int64
 19  Dist_Sum_INTER   2910 non-null   float64
 20  Dist_Mean_INTER  2910 non-null   float64
 21  Dist_Max_INTER   2910 non-null   float64
 22  NROCIUDADES      2910 non-null   int64
 23  Dist_Mean_NAL    2910 non-null   float64
 24  Dist_HOY         2910 non-null   float64
 25  Dist_sum_NAL     2910 non-null   float64
dtypes: float64(13), int64(8), object(5)
memory usage: 613.8+ KB
```

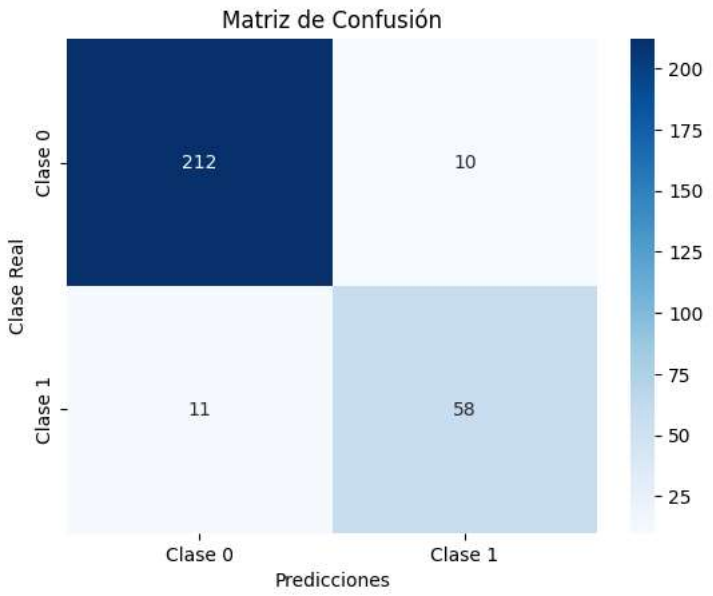
MODELO DE ML

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score


def split_data(data, target_column='FRAUDE', test_size=0.1, random_state=42):
    """Dividir el conjunto de datos en entrenamiento y prueba."""
    X = data.drop(target_column, axis=1)
    y = data[target_column]
    return train_test_split(X, y, test_size=test_size, random_state=random_state)

def build_preprocessor(numeric_features, categorical_features):
    """Construir el preprocesador utilizando ColumnTransformer."""
    numeric_transformer = Pipeline(steps=[
        ('scaler', StandardScaler())
    ])

    categorical_transformer = Pipeline(steps=[
        ('onehot', OneHotEncoder(handle_unknown='ignore'))
    ])

    return ColumnTransformer(
        transformers=[
            ('num', numeric_transformer, numeric_features),
            ('cat', categorical_transformer, categorical_features)
        ])

def build_model(preprocessor, classifier=LogisticRegression()):
    """Construir el modelo de clasificación."""
    return Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', classifier)
    ])

def train_model(model, X_train, y_train):
    """Entrenar el modelo."""
    model.fit(X_train, y_train)

def evaluate_model(model, X_test, y_test):
    """Evaluar el modelo y devolver la precisión."""
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return accuracy


# Dividir datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = split_data(data)

# Definir columnas numéricas y categóricas
numeric_features = ['VALOR', 'HORA_AUX', 'Dist_max_NAL', 'EDAD', 'INGRESOS', 'EGRESOS', 'NROPAISES', 'Dist_Sum_INTER', 'Dist_Mean_INTER', 'Dist_Max_INTER',
```

```
categorical_features = ['Canal1', 'COD_PAIS', 'CANAL', 'DIASEM', 'DIAMES', 'FECHA_VIN', 'OFICINA_VIN', 'SEXO', 'SEGMENTO']

# Construir preprocesador
preprocessor = build_preprocessor(numeric_features, categorical_features)

# Construir y entrenar el modelo
model = build_model(preprocessor)
train_model(model, X_train, y_train)

# Evaluar el modelo
accuracy = evaluate_model(model, X_test, y_test)
print(f'Precisión del modelo: {accuracy}')
```

```
Precisión del modelo: 0.9278350515463918
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, y_pred)


labels = ['Clase 0', 'Clase 1']
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
plt.title('Matriz de Confusión')
plt.xlabel('Predicciones')
plt.ylabel('Clase Real')
plt.show()
# Verdaderos Negativos : 212 casos fueron correctamente clasificados como la clase 0.
# Falsos Positivos : 10 casos fueron incorrectamente clasificados como la clase 1.
# Falsos Negativos : 11 casos fueron incorrectamente clasificados como la clase 0.
# Verdaderos Positivos : 58 casos fueron correctamente clasificados como la clase 1.
```



TEST

```
test_data = pd.read_csv('/test.csv')

test_data.rename(columns={'Dist_max_COL': 'Dist_max_NAL'}, inplace=True)
```

```
columns_to_select = ['id', 'VALOR', 'HORA_AUX', 'Dist_max_NAL', 'Canal1', 'FECHA', 'COD_PAIS', 'CANAL', 'DIASEM', 'DIAMES', 'FECHA_VIN', 'OFICINA_VIN', 'S
new_test_data = test_data[columns_to_select]
```

```
#EXISTEN INCOHERENCIAS ENTRE LOS DATOS DE ENTRENAMIENTO Y TEST

condition1 = new_test_data['NROCIUDADES'] == '1'
new_test_data.loc[condition1, ['Dist_Mean_NAL', 'Dist_sum_NAL', 'Dist_max_NAL']] = 0
condition2 = new_test_data['NROPAISES'] == '1'
new_test_data.loc[condition1, ['Dist_Sum_INTER', 'Dist_Mean_INTER', 'Dist_Max_INTER']] = 0
```

```
new_test_data.fillna(0, inplace=True)
new_test_data
```

```
<ipython-input-183-1885abd72c7c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
  new_test_data.fillna(0, inplace=True)
```

	id	VALOR	HORA_AUX	Dist_max_NAL	Canal1	FECHA	COD_PAIS	CANAL	DIASEM
0	98523068	42230.09	18	1.00	POS	20150515	US	POS	5
1	300237898	143202.65	20	614.04	POS	20150506	US	MCI	3
2	943273308	243591.25	2	286.84	ATM_INT	20150517	EC	ATM_INT	0
3	951645809	238267.40	20	1.00	ATM_INT	20150508	EC	ATM_INT	5
4	963797516	490403.58	13	1.00	ATM_INT	20150501	US	ATM_INT	5
...	...	...	...	...	...	...	...	...	...
95	9970518152	531534.03	13	340.09	POS	20150501	US	POS	5
96	9971748725	52035.08	11	28.59	POS	20150503	AW	POS	0
97	9979565282	18309.04	23	61.45	POS	20150515	US	POS	5
98	9979718478	496906.75	20	733.11	ATM_INT	20150516	US	ATM_INT	6
99	9998668320	192825.50	20	337.29	POS	20150515	US	MCI	5

100 rows × 25 columns

```
y_test_processed = model.predict(new_test_data)
y_test_processed
```

```
array([0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0])
```

```
data_test = pd.read_csv('/test.csv')
data_test['FRAUDE'] = y_test_processed
data_test.to_csv('/salida.csv', index=False)
```