

# Fraud Detection Using ML and Web App

July 19, 2025

```
[1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import warnings
warnings.filterwarnings('ignore')
```

```
/Users/hinalpanchal/opt/anaconda3/lib/python3.9/site-
packages/scipy/__init__.py:155: UserWarning: A NumPy version >=1.18.5 and
<1.25.0 is required for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")
```

```
[2]: fraud_data = pd.read_csv('Fraud_Detection.csv')

fraud_data.head()
```

```
[2]:
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	\
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	M1979787155	0.0	0.0	0	0
1	M2044282225	0.0	0.0	0	0
2	C553264065	0.0	0.0	1	0
3	C38997010	21182.0	0.0	1	0
4	M1230701703	0.0	0.0	0	0

```
[3]: fraud_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
#   Column      Dtype
---  -
0   step        int64
```

```

1  type           object
2  amount         float64
3  nameOrig       object
4  oldbalanceOrg  float64
5  newbalanceOrig float64
6  nameDest       object
7  oldbalanceDest float64
8  newbalanceDest float64
9  isFraud        int64
10 isFlaggedFraud int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB

```

```
[4]: fraud_data.isna().sum()
```

```

[4]: step           0
     type           0
     amount         0
     nameOrig       0
     oldbalanceOrg  0
     newbalanceOrig 0
     nameDest       0
     oldbalanceDest 0
     newbalanceDest 0
     isFraud        0
     isFlaggedFraud 0
     dtype: int64

```

```
[5]: fraud_data.columns
```

```

[5]: Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
          'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
          'isFlaggedFraud'],
          dtype='object')

```

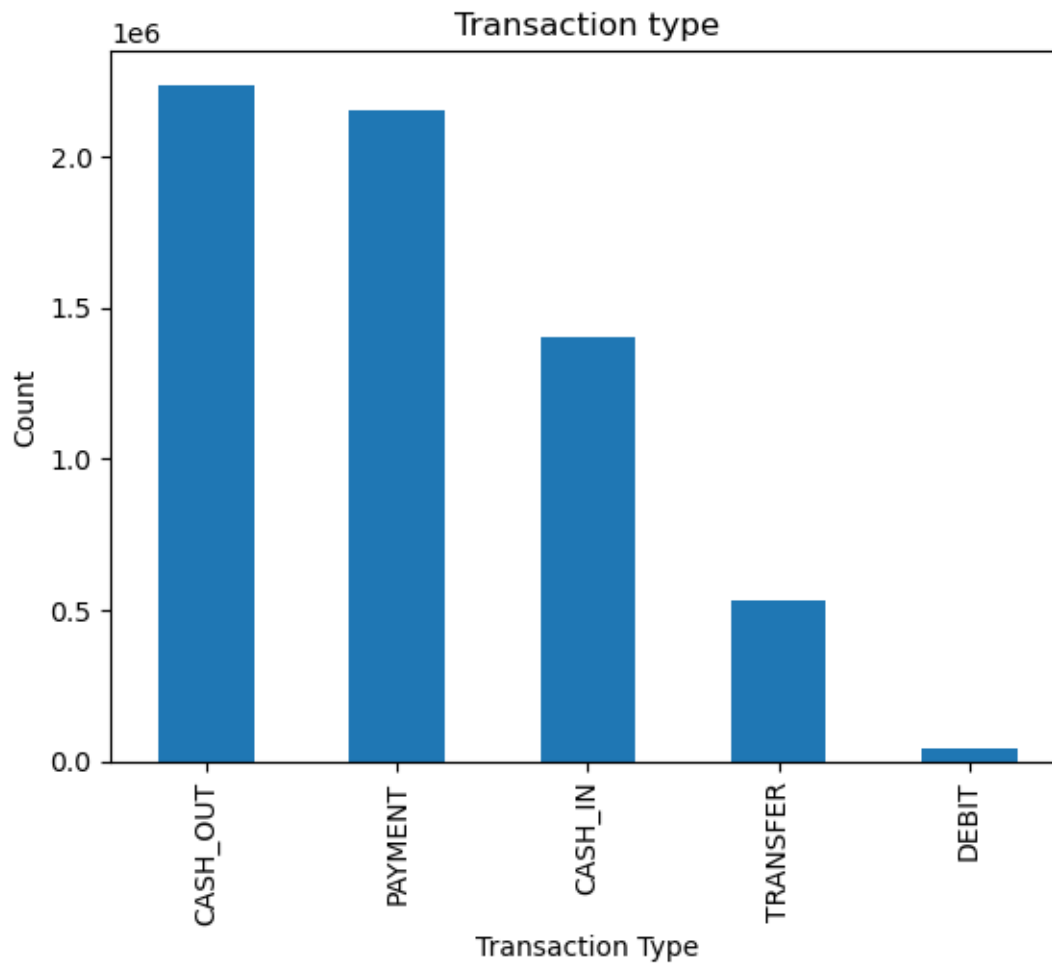
## 1 EDA

```

[6]: # Visualize the TYPE of transction

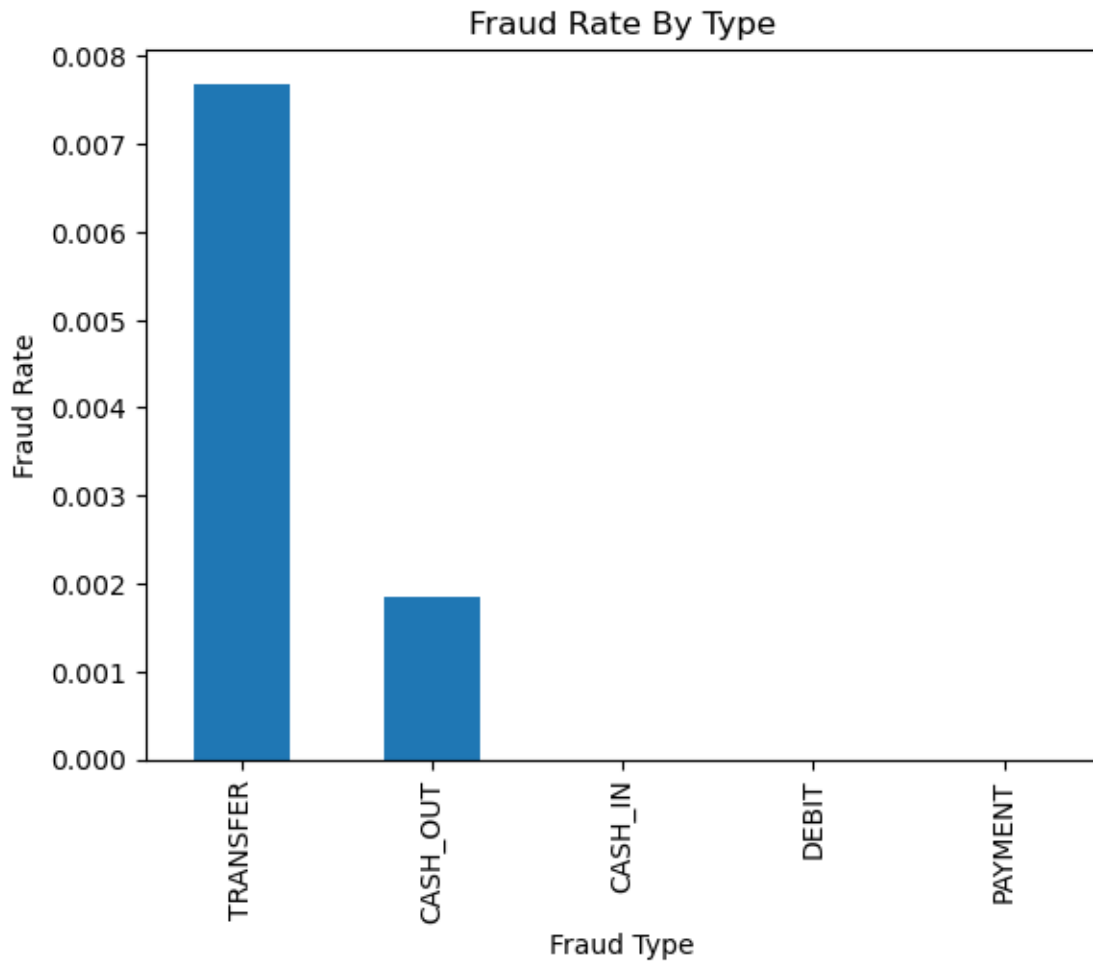
fraud_data['type'].value_counts().plot(kind='bar',title='Transaction type')
plt.xlabel('Transaction Type')
plt.ylabel('Count')
plt.show()

```



```
[7]: # Fraud rate by type

fraud_by_type= fraud_data.groupby('type')['isFraud'].mean().
    ↪sort_values(ascending=False)
fraud_by_type.plot(kind='bar',title='Fraud Rate By Type')
plt.xlabel('Fraud Type')
plt.ylabel('Fraud Rate')
plt.show()
```



```
[8]: fraud_data['balanceDiffOrgi'] =_
      ↪ fraud_data['oldbalanceOrg']-fraud_data['newbalanceOrg']
fraud_data['balanceDiffDest'] =_
      ↪ fraud_data['newbalanceDest']-fraud_data['oldbalanceDest']
```

```
[9]: (fraud_data['balanceDiffOrgi']< 0).sum()
```

```
[9]: 1399253
```

```
[10]: (fraud_data['balanceDiffDest']< 0).sum()
```

```
[10]: 1238864
```

```
[11]: fraud_data.head()
```

```
[11]:   step    type  amount  nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1  PAYMENT  9839.64  C1231006815         170136.0         160296.36
```

1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72
2	1	TRANSFER	181.00	C1305486145	181.0	0.00
3	1	CASH_OUT	181.00	C840083671	181.0	0.00
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	\
0	M1979787155	0.0	0.0	0	0	
1	M2044282225	0.0	0.0	0	0	
2	C553264065	0.0	0.0	1	0	
3	C38997010	21182.0	0.0	1	0	
4	M1230701703	0.0	0.0	0	0	

	balanceDiffOrgi	balanceDiffDest
0	9839.64	0.0
1	1864.28	0.0
2	181.00	0.0
3	181.00	-21182.0
4	11668.14	0.0

```
[12]: fraud_data.drop(columns='step', inplace=True)
```

```
[13]: fraud_data.head()
```

```
[13]:
```

	type	amount	nameOrig	oldbalanceOrg	newbalanceOrg	\
0	PAYMENT	9839.64	C1231006815	170136.0	160296.36	
1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	
2	TRANSFER	181.00	C1305486145	181.0	0.00	
3	CASH_OUT	181.00	C840083671	181.0	0.00	
4	PAYMENT	11668.14	C2048537720	41554.0	29885.86	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	\
0	M1979787155	0.0	0.0	0	0	
1	M2044282225	0.0	0.0	0	0	
2	C553264065	0.0	0.0	1	0	
3	C38997010	21182.0	0.0	1	0	
4	M1230701703	0.0	0.0	0	0	

	balanceDiffOrgi	balanceDiffDest
0	9839.64	0.0
1	1864.28	0.0
2	181.00	0.0
3	181.00	-21182.0
4	11668.14	0.0

```
[14]: top_senders = fraud_data['nameOrig'].value_counts().head(10)
top_senders
```

```
[14]: C1902386530    3
      C363736674    3
      C545315117    3
      C724452879    3
      C1784010646    3
      C1677795071    3
      C1462946854    3
      C1999539787    3
      C2098525306    3
      C400299098     3
      Name: nameOrig, dtype: int64
```

```
[15]: top_receiver = fraud_data['nameDest'].value_counts().head(10)
      top_receiver
```

```
[15]: C1286084959    113
      C985934102     109
      C665576141     105
      C2083562754    102
      C248609774     101
      C1590550415    101
      C451111351      99
      C1789550256     99
      C1360767589     98
      C1023714065     97
      Name: nameDest, dtype: int64
```

```
[16]: fraud_users = fraud_data[fraud_data['isFraud']==1]['nameOrig'].value_counts().
      ↪head(10)
      fraud_users
```

```
[16]: C1305486145     1
      C755286039     1
      C973279667     1
      C258213312     1
      C1640703547     1
      C1127265876     1
      C317779855     1
      C1064034527     1
      C1141104763     1
      C1966863341     1
      Name: nameOrig, dtype: int64
```

```
[17]: fraud_types = fraud_data[fraud_data['type'].isin(['TRANSFER', 'CASH_OUT'])]
      fraud_types
```

```
[17]:
```

	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	\
2	TRANSFER	181.00	C1305486145	181.00	0.0	
3	CASH_OUT	181.00	C840083671	181.00	0.0	
15	CASH_OUT	229133.94	C905080434	15325.00	0.0	
19	TRANSFER	215310.30	C1670993182	705.00	0.0	
24	TRANSFER	311685.89	C1984094095	10835.00	0.0	
...	...	...	...	...	...	
6362615	CASH_OUT	339682.13	C786484425	339682.13	0.0	
6362616	TRANSFER	6311409.28	C1529008245	6311409.28	0.0	
6362617	CASH_OUT	6311409.28	C1162922333	6311409.28	0.0	
6362618	TRANSFER	850002.52	C1685995037	850002.52	0.0	
6362619	CASH_OUT	850002.52	C1280323807	850002.52	0.0	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	\
2	C553264065	0.00	0.00	1	0	
3	C38997010	21182.00	0.00	1	0	
15	C476402209	5083.00	51513.44	0	0	
19	C1100439041	22425.00	0.00	0	0	
24	C932583850	6267.00	2719172.89	0	0	
...	...	...	...	...	...	
6362615	C776919290	0.00	339682.13	1	0	
6362616	C1881841831	0.00	0.00	1	0	
6362617	C1365125890	68488.84	6379898.11	1	0	
6362618	C2080388513	0.00	0.00	1	0	
6362619	C873221189	6510099.11	7360101.63	1	0	

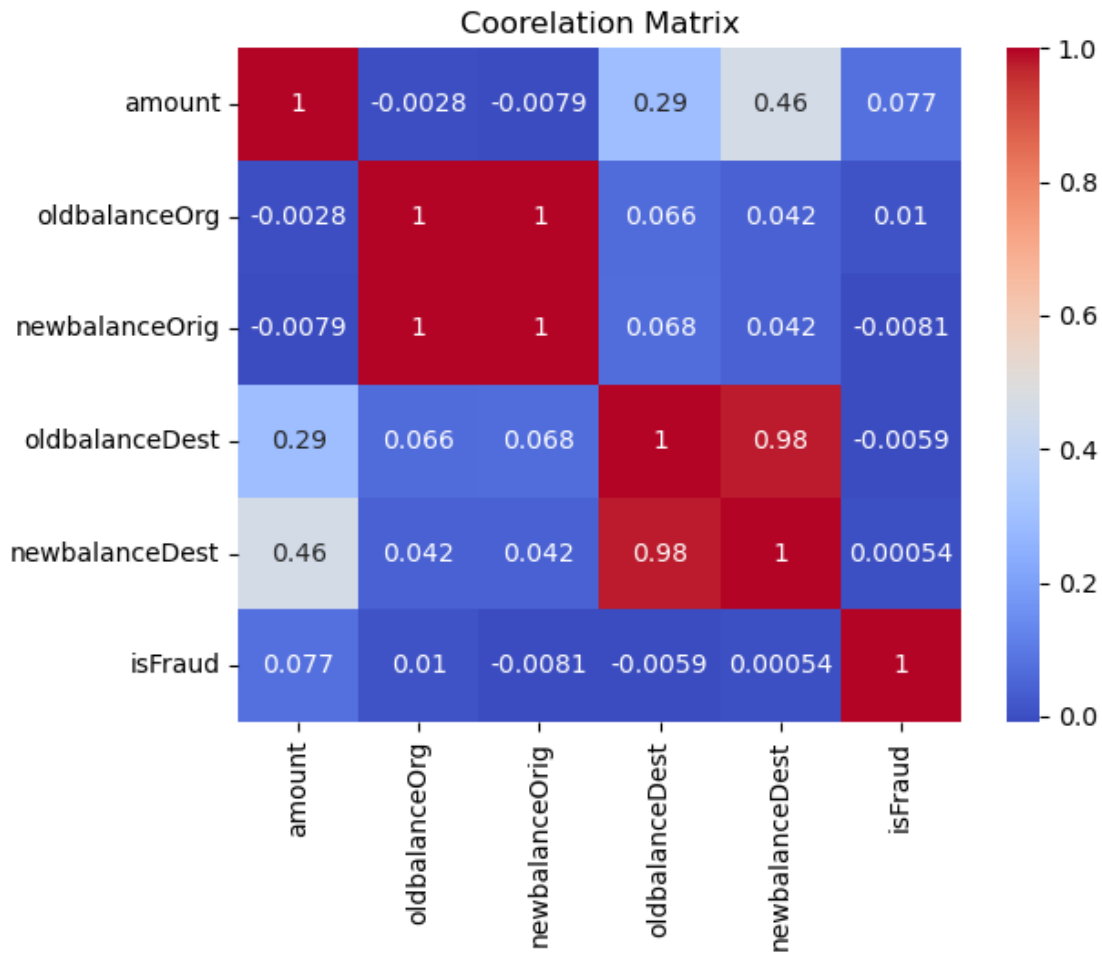
	balanceDiffOrgi	balanceDiffDest
2	181.00	0.00
3	181.00	-21182.00
15	15325.00	46430.44
19	705.00	-22425.00
24	10835.00	2712905.89
...	...	...
6362615	339682.13	339682.13
6362616	6311409.28	0.00
6362617	6311409.28	6311409.27
6362618	850002.52	0.00
6362619	850002.52	850002.52

[2770409 rows x 12 columns]

```
[18]: corr = fraud_data[['amount', 'oldbalanceOrg', 'newbalanceOrig',
↪ 'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud']].corr()
```

```
[19]: sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Coorelation Matrix')
```

```
[19]: Text(0.5, 1.0, 'Coorelation Matrix')
```



## 2 Machine Learning Implementation

```
[20]: from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
[21]: fraud_data.columns
```



```
[21]: Index(['type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
          'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
          'isFlaggedFraud', 'balanceDiffOrgi', 'balanceDiffDest'],
          dtype='object')
```

```
[22]: fraud_model = fraud_data.drop(['nameOrig', 'nameDest', 'isFlaggedFraud'], axis=1)
```

```
[23]: fraud_model.columns
```

```
[23]: Index(['type', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
          'newbalanceDest', 'isFraud', 'balanceDiffOrgi', 'balanceDiffDest'],
          dtype='object')
```

```
[24]: fraud_model.head(5)
```

```
[24]:
```

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	\
0	PAYMENT	9839.64	170136.0	160296.36	0.0	
1	PAYMENT	1864.28	21249.0	19384.72	0.0	
2	TRANSFER	181.00	181.0	0.00	0.0	
3	CASH_OUT	181.00	181.0	0.00	21182.0	
4	PAYMENT	11668.14	41554.0	29885.86	0.0	

	newbalanceDest	isFraud	balanceDiffOrgi	balanceDiffDest
0	0.0	0	9839.64	0.0
1	0.0	0	1864.28	0.0
2	0.0	1	181.00	0.0
3	0.0	1	181.00	-21182.0
4	0.0	0	11668.14	0.0

```
[25]: # Define features
categorical = ['type']
numeric = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
          ↪ 'newbalanceDest']
X = fraud_model.drop('isFraud', axis=1)
y = fraud_model['isFraud']

# Define transformers
preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(), numeric),
    ('cat', OneHotEncoder(), categorical)
])

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
          ↪ test_size=0.3, random_state=42)
```

```
[26]: preprocessor = ColumnTransformer(
      transformers= [
          ('num',StandardScaler(), numeric),
          ('cat',OneHotEncoder(drop='first'), categorical)
      ],
      remainder='drop'
  )
```

```
[27]: pipeline =Pipeline([
      ('prep',preprocessor),

      ('clf',LogisticRegression(class_weight='balanced',max_iter=1000))
  ])
```

```
[28]: pipeline.fit(X_train,y_train)
```

```
[28]: Pipeline(steps=[('prep',
                       ColumnTransformer(transformers=[('num', StandardScaler(),
                                                         ['amount', 'oldbalanceOrig',
                                                         'newbalanceOrig',
                                                         'oldbalanceDest',
                                                         'newbalanceDest']),
                                                         ('cat',
                                                          OneHotEncoder(drop='first'),
                                                          ['type'])])),
                       ('clf',
                        LogisticRegression(class_weight='balanced', max_iter=1000))])
```

```
[29]: y_pred = pipeline.predict(X_test)
```

```
[30]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.95	0.97	1906322
1	0.02	0.94	0.04	2464
accuracy			0.95	1908786
macro avg	0.51	0.94	0.51	1908786
weighted avg	1.00	0.95	0.97	1908786

```
[31]: print(confusion_matrix(y_test,y_pred))
```

```
[[1805566 100756]
 [   151   2313]]
```

```
[32]: import joblib  
      joblib.dump(pipeline, 'fraud_detection_pipeline.pkl')
```

```
[32]: ['fraud_detection_pipeline.pkl']
```

```
[ ]:
```