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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('onlinefraud.csv')

df.head()

   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

df.info

<bound method DataFrame.info of          step      type      amount
nameOrig  oldbalanceOrg  \
0           1   PAYMENT     9839.64  C1231006815      170136.00
1           1   PAYMENT     1864.28  C1666544295       21249.00
2           1  TRANSFER      181.00  C1305486145         181.00
3           1  CASH_OUT      181.00   C840083671         181.00
4           1   PAYMENT    11668.14  C2048537720       41554.00
```

```
...       ...       ...        ...          ...            ...
6362615   743  CASH_OUT   339682.13   C786484425      339682.13
6362616   743  TRANSFER  6311409.28  C1529008245     6311409.28
6362617   743  CASH_OUT  6311409.28  C1162922333     6311409.28
6362618   743  TRANSFER   850002.52  C1685995037      850002.52
6362619   743  CASH_OUT   850002.52  C1280323807      850002.52

         newbalanceOrig     nameDest  oldbalanceDest  newbalanceDest
isFraud  \
0             160296.36  M1979787155            0.00            0.00
0
1              19384.72  M2044282225            0.00            0.00
0
2                  0.00   C553264065            0.00            0.00
1
3                  0.00    C38997010        21182.00            0.00
1
4              29885.86  M1230701703            0.00            0.00
0
...                 ...          ...             ...             ...
...
6362615            0.00   C776919290            0.00       339682.13
1
6362616            0.00  C1881841831            0.00            0.00
1
6362617            0.00  C1365125890        68488.84      6379898.11
1
6362618            0.00  C2080388513            0.00            0.00
1
6362619            0.00   C873221189      6510099.11      7360101.63
1

         isFlaggedFraud
0                     0
1                     0
2                     0
3                     0
4                     0
...                 ...
6362615               0
6362616               0
6362617               0
6362618               0
6362619               0

[6362620 rows x 11 columns]>
```

```
df.shape
```

```
(6362620, 11)
```

```
df['step'].value_counts()
```

```
step
19     51352
18     49579
187    49083
235    47491
307    46968
       ...  
432        4
706        4
693        4
112        2
662        2
Name: count, Length: 743, dtype: int64
```

```
df['type'].value_counts()
```

```
type
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: count, dtype: int64
```

```
df['nameOrig'].value_counts()
```

```
nameOrig
C1902386530    3
C363736674     3
C545315117     3
C724452879     3
C1784010646    3
              ..
C98968405      1
C720209255     1
C1567523029    1
C644777639     1
C1280323807    1
Name: count, Length: 6353307, dtype: int64
```

```
df['nameDest'].value_counts()
```

```
nameDest
C1286084959    113
C985934102     109
C665576141     105
C2083562754    102
C248609774     101
              ... 
```

```
M1470027725      1
M1330329251      1
M1784358659      1
M2081431099      1
C2080388513      1
Name: count, Length: 2722362, dtype: int64
```

```
df['isFraud'].value_counts()
```

```
isFraud
No Fraud    6354407
Fraud          8213
Name: count, dtype: int64
```

```
df.isnull().sum()
```

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

```
df.describe()
```

```
               step          type        amount  oldbalanceOrg  \
count  6.362620e+06  6.362620e+06  6.362620e+06   6.362620e+06
mean   2.433972e+02  2.055307e+00  1.798619e+05   8.338831e+05
std    1.423320e+02  9.808966e-01  6.038582e+05   2.888243e+06
min    1.000000e+00  1.000000e+00  0.000000e+00   0.000000e+00
25%    1.560000e+02  1.000000e+00  1.338957e+04   0.000000e+00
50%    2.390000e+02  2.000000e+00  7.487194e+04   1.420800e+04
75%    3.350000e+02  3.000000e+00  2.087215e+05   1.073152e+05
max    7.430000e+02  5.000000e+00  9.244552e+07   5.958504e+07

       newbalanceOrig  oldbalanceDest  newbalanceDest  isFlaggedFraud

count    6.362620e+06    6.362620e+06    6.362620e+06    6.362620e+06

mean     8.551137e+05    1.100702e+06    1.224996e+06    2.514687e-06

std      2.924049e+06    3.399180e+06    3.674129e+06    1.585775e-03

min      0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00
```

```
25%      0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00

50%      0.000000e+00    1.327057e+05    2.146614e+05    0.000000e+00

75%      1.442584e+05    9.430367e+05    1.111909e+06    0.000000e+00

max      4.958504e+07    3.560159e+08    3.561793e+08    1.000000e+00
```

```
numeric_df = df.select_dtypes(include=['float64', 'int64'])
correlation = numeric_df.corr()
print(correlation["isFraud"].sort_values(ascending=False))
```

```
isFraud           1.000000
amount            0.076688
isFlaggedFraud    0.044109
step              0.031578
oldbalanceOrg     0.010154
newbalanceDest    0.000535
oldbalanceDest   -0.005885
newbalanceOrig   -0.008148
Name: isFraud, dtype: float64
```

```
df["type"] = df["type"].replace({"CASH_OUT": 1, "PAYMENT": 2,
"CASH_IN": 3, "TRANSFER": 4, "DEBIT": 5})
df["isFraud"] = df["isFraud"].replace({0: "No Fraud", 1: "Fraud"})
print(df.head())
```

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

      nameDest  oldbalanceDest  newbalanceDest   isFraud
isFlaggedFraud
0  M1979787155             0.0             0.0  No Fraud
0
1  M2044282225             0.0             0.0  No Fraud
0
2   C553264065             0.0             0.0     Fraud
0
3    C38997010         21182.0             0.0     Fraud
0
4  M1230701703             0.0             0.0  No Fraud
0
```

Splitting the data

```
x = np.array(df[["type", "amount", "oldbalanceOrg",
"newbalanceOrig"]])
y = np.array(df[["isFraud"]])
```

Training the model

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.10,
random_state=42)

model_LReg = LogisticRegression()
model_LReg.fit(xtrain, ytrain)
model_LReg.score(xtrain, ytrain)

0.9994926967542023

model_LReg.score(xtest, ytest)

0.9995049209287997

model_DTC = DecisionTreeClassifier()
model_DTC.fit(xtrain, ytrain)
print(model_DTC.score(xtest, ytest))

0.9997438162266488
```

Prediction

```
#features = [type, amount, oldbalanceOrg, newbalanceOrig]
features = np.array([[4, 9000.60, 9000.60, 0.0]])
print(model.predict(features))

['Fraud']

features = np.array([[2,9839.64,170136.0,160296.36]])
print(model.predict(features))

['No Fraud']
```