```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('onlinefraud.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix, recall_score, f1_score

import scipy.stats as st

import warnings
warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)


df = pd.read_csv("onlinefraud.csv")
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDes
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M197978715
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M204428222
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C5532640
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C3899701
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M123070170

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14247 entries, 0 to 14246
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   step            14247 non-null  int64
 1   type            14247 non-null  object
 2   amount          14247 non-null  float64
 3   nameOrig        14247 non-null  object
 4   oldbalanceOrg   14246 non-null  float64
 5   newbalanceOrig  14246 non-null  float64
 6   nameDest        14246 non-null  object
 7   oldbalanceDest  14246 non-null  float64
 8   newbalanceDest  14246 non-null  float64
 9   isFraud         14246 non-null  float64
 10  isFlaggedFraud  14246 non-null  float64
dtypes: float64(7), int64(1), object(3)
memory usage: 1.2+ MB
```

```
df.describe()
```

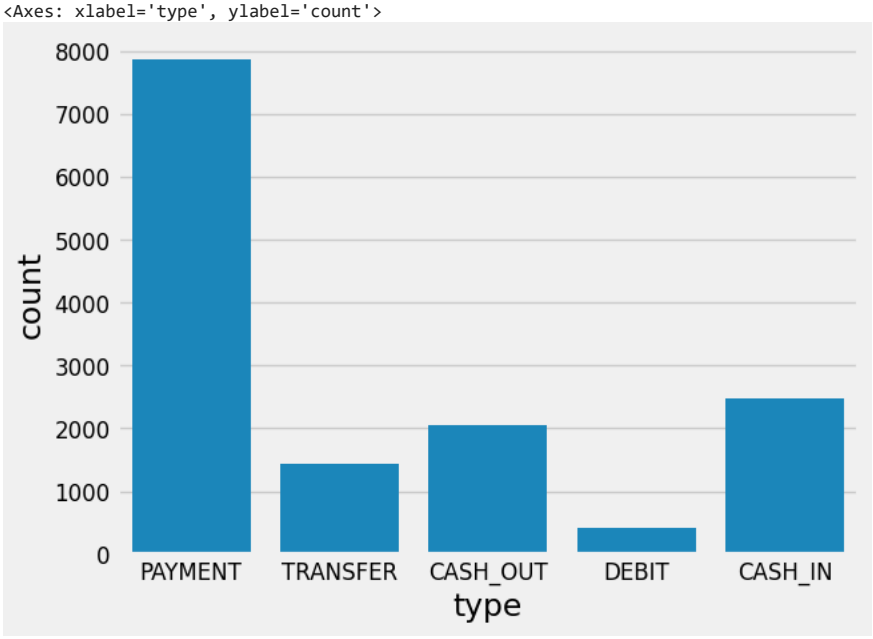
	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newb
count	14247.000000	1.424700e+04	1.424600e+04	1.424600e+04	1.424600e+04	1
mean	5.037131	1.118848e+05	7.937369e+05	8.107136e+05	8.411336e+05	1
std	2.463112	2.805152e+05	2.016693e+06	2.059984e+06	2.528174e+06	3
min	1.000000	2.390000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
25%	2.000000	4.585130e+03	0.000000e+00	0.000000e+00	0.000000e+00	0
50%	6.000000	1.286936e+04	2.041500e+04	8.216325e+03	0.000000e+00	0
75%	7.000000	1.218917e+05	1.384956e+05	1.294173e+05	2.641427e+05	2
max	8.000000	1.000000e+07	1.293042e+07	1.301050e+07	2.093759e+07	2

```
df = df.iloc[:,:-1]

obj = (df.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:", len(object_cols))

int_ = (df.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:", len(num_cols))

fl = (df.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:", len(fl_cols))
```

```
Categorical variables: 3
Integer variables: 1
Float variables: 5
```

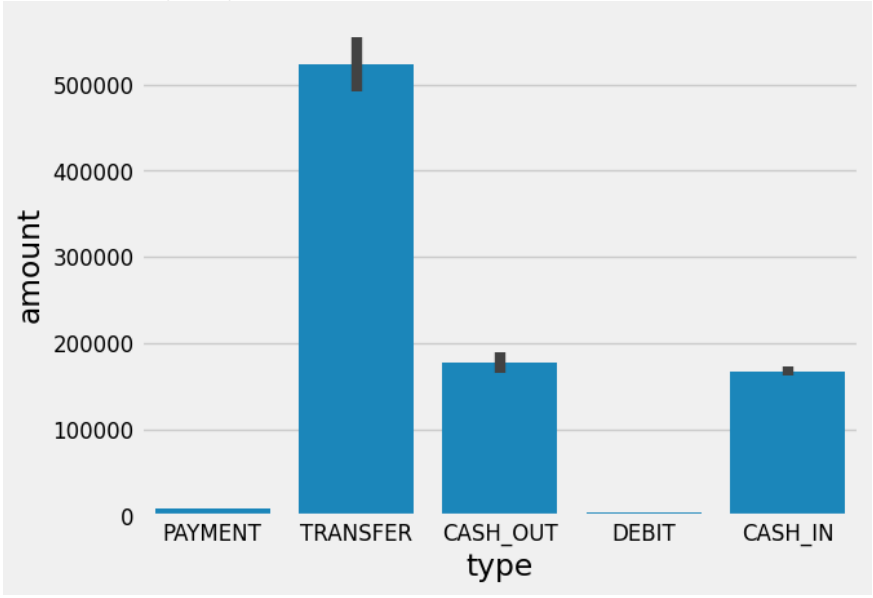
```
sns.countplot(x='type', data=df)
```

```
<Axes: xlabel='type', ylabel='count'>
```




```
sns.barplot(x='type', y='amount', data=df)
```

```
<Axes: xlabel='type', ylabel='amount'>
```

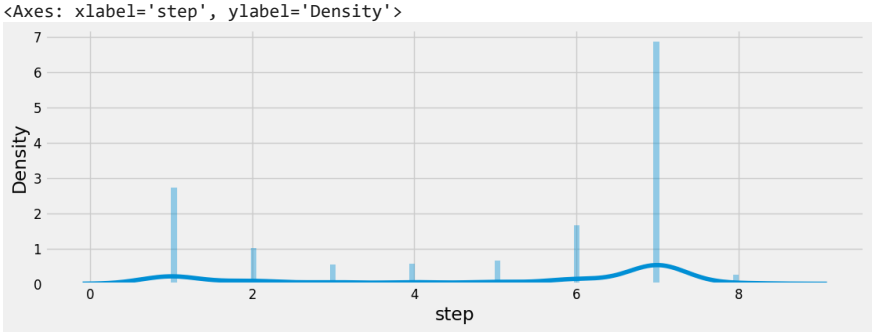



```
df = pd.read_csv("onlinefraud.csv")
df['isFraud'].value_counts()
```

```
0.0    476923
1.0       224
Name: isFraud, dtype: int64
```

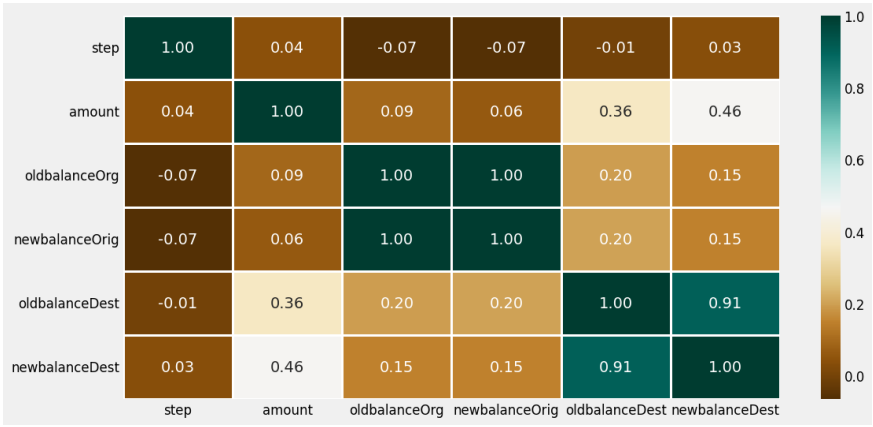
```
plt.figure(figsize=(12, 4))
sns.distplot(df['step'], bins=100)
```

```
<Axes: xlabel='step', ylabel='Density'>
```




```
numeric_df = df.select_dtypes(include=['number'])
plt.figure(figsize=(12, 6))
sns.heatmap(numeric_df.corr(), cmap='BrBG', fmt='.2f', linewidths=2, annot=True)
```

```
<Axes: >
```




```
type_new = pd.get_dummies(df['type'], drop_first=True)
df_new = pd.concat([df, type_new], axis=1)
df_new.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDes
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M197978715
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M204428222
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C55326406
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C3899701
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M123070170

```
X = df.drop(['isFraud', 'type', 'nameOrig', 'nameDest'], axis=1)
y = df['isFraud']
X.shape, y.shape
```

```
((545326, 7), (545326,))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

```
#Train Test split - By default train_test_split does STRATIFIED split based on label (y-value).
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
XGB = XGBClassifier()
XGB.fit(X_train, y_train)
```

```
XGB.score(X_train,y_train)
```

```
0.9999083116774248
```

```
# Check for NaN values in y_test
nan_indices = np.isnan(y_test)
# Print the indices of NaN values
print(np.where(nan_indices))
# Drop rows with NaN values in y_test
X_test = X_test[~nan_indices]
y_test = y_test[~nan_indices]
```

```
(array([50370]),)
```

```
XGB.score(X_test,y_test)
```

```
0.999724933831305
```

```
LR = LogisticRegression()
LR.fit(X_train,y_train)
```

```
LR.score(X_train,y_train)
```

```
0.9996044303797469
```

```
LR.score(X_test,y_test)
```

```
0.9996760331790925
```

```
from sklearn.impute import SimpleImputer

#Removing NaN rows
nan_indices = np.isnan(y)
X = X[~nan_indices]
y = y[~nan_indices]

#Imputing NaN values
imputer = SimpleImputer(strategy='mean')
y = imputer.fit_transform(y.values.reshape(-1, 1)).flatten()

# Now you can proceed with predicting and constructing the confusion matrix
y_pred = XGB.predict(X)

cm = confusion_matrix(y, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```