In [193…

```python
import  pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import OneHotEncoder
```

In [203…

```python
df = pd.read_csv("fraud.csv")
```

In [204…

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column          Non-Null Count    Dtype  
---  ------          --------------    -----  
 0   step            1048575 non-null  int64  
 1   type            1048575 non-null  object 
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object 
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object 
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64  
 10  isFlaggedFraud  1048575 non-null  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 88.0+ MB
```

In [205…

```python
df.head()
```

Out[205]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

In [206…

```python
df.shape
```

Out[206]:

```
(1048575, 11)
```

In [207…

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

Out[207]:

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

In [208…

```python
df.corr()
```

```
C:\Users\tanmay pandey\AppData\Local\Temp\ipykernel_1232\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will def
ault to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[208]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
step	1.000000	-0.025996	-0.006780	-0.007180	-0.002251	-0.019503	0.045030	NaN
amount	-0.025996	1.000000	0.004864	-0.001133	0.215558	0.311936	0.128862	NaN
oldbalanceOrg	-0.006780	0.004864	1.000000	0.999047	0.093305	0.064049	0.003829	NaN
newbalanceOrig	-0.007180	-0.001133	0.999047	1.000000	0.095182	0.063725	-0.009438	NaN
oldbalanceDest	-0.002251	0.215558	0.093305	0.095182	1.000000	0.978403	-0.007552	NaN
newbalanceDest	-0.019503	0.311936	0.064049	0.063725	0.978403	1.000000	-0.000495	NaN
isFraud	0.045030	0.128862	0.003829	-0.009438	-0.007552	-0.000495	1.000000	NaN
isFlaggedFraud	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [209…

```python
df["type"].unique()
```

Out[209]:

```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

In [210…

```python
df["dif_balance_org"] = df["newbalanceOrig"] - df["oldbalanceOrg"]

df["dif_balance_dest"] = df["newbalanceDest"] - df["oldbalanceDest"]
```

In [211…

```python
df.drop(df.iloc[:,[4,5,7,8,10]], axis = 1 , inplace = True )
```

In [212…

```python
df.head()
```

Out[212]:

	step	type	amount	nameOrig	nameDest	isFraud	dif_balance_org	dif_balance_dest
0	1	PAYMENT	9839.64	C1231006815	M1979787155	0	-9839.64	0.0
1	1	PAYMENT	1864.28	C1666544295	M2044282225	0	-1864.28	0.0
2	1	TRANSFER	181.00	C1305486145	C553264065	1	-181.00	0.0
3	1	CASH_OUT	181.00	C840083671	C38997010	1	-181.00	-21182.0
4	1	PAYMENT	11668.14	C2048537720	M1230701703	0	-11668.14	0.0

In [213…

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column            Non-Null Count    Dtype  
---  ------            --------------    -----  
 0   step              1048575 non-null  int64  
 1   type              1048575 non-null  object 
 2   amount            1048575 non-null  float64
 3   nameOrig          1048575 non-null  object 
 4   nameDest          1048575 non-null  object 
 5   isFraud           1048575 non-null  int64  
 6   dif_balance_org   1048575 non-null  float64
 7   dif_balance_dest  1048575 non-null  float64
dtypes: float64(3), int64(2), object(3)
memory usage: 64.0+ MB
```

In [214…

```python
df.corr()
```

```
C:\Users\tanmay pandey\AppData\Local\Temp\ipykernel_1232\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will def
ault to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[214]:

	step	amount	isFraud	dif_balance_org	dif_balance_dest
step	1.000000	-0.025996	0.045030	-0.010708	-0.083239
amount	-0.025996	1.000000	0.128862	-0.131801	0.513277
isFraud	0.045030	0.128862	1.000000	-0.293467	0.032034
dif_balance_org	-0.010708	-0.131801	-0.293467	1.000000	-0.256157
dif_balance_dest	-0.083239	0.513277	0.032034	-0.256157	1.000000

In [215…

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

Out[215]:

```
step                0
type                0
amount              0
nameOrig            0
nameDest            0
isFraud             0
dif_balance_org     0
dif_balance_dest    0
dtype: int64
```

In [216…

```python
onehot_encoder = OneHotEncoder(sparse=False)

# Fit and transform the 'Category' column
encoded_data = onehot_encoder.fit_transform(df[['type']])

# Get the feature names for the one-hot encoded columns
feature_names = onehot_encoder.get_feature_names_out(input_features=['type'])

# Create a DataFrame from the encoded data with proper column names
df1 = pd.DataFrame(encoded_data, columns=feature_names)
```

```
C:\Users\tanmay pandey\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessing\_encoders.py:972: FutureWarning: `sparse` was renamed to `sparse_output` in version
1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [217…

```python
df = pd.concat([df, df1] , axis = 1)
```

In [218…

```python
df.head()
```

Out[218]:

	step	type	amount	nameOrig	nameDest	isFraud	dif_balance_org	dif_balance_dest	type_CASH_IN	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER
0	1	PAYMENT	9839.64	C1231006815	M1979787155	0	-9839.64	0.0	0.0	0.0	0.0	1.0	0.0
1	1	PAYMENT	1864.28	C1666544295	M2044282225	0	-1864.28	0.0	0.0	0.0	0.0	1.0	0.0
2	1	TRANSFER	181.00	C1305486145	C553264065	1	-181.00	0.0	0.0	0.0	0.0	0.0	1.0
3	1	CASH_OUT	181.00	C840083671	C38997010	1	-181.00	-21182.0	0.0	1.0	0.0	0.0	0.0
4	1	PAYMENT	11668.14	C2048537720	M1230701703	0	-11668.14	0.0	0.0	0.0	0.0	1.0	0.0

In [219…

```python
df.drop(df.iloc[:,[1,3,4]], axis = 1 , inplace = True)
```

In [220…

```python
df.head()
```

Out[220]:

	step	amount	isFraud	dif_balance_org	dif_balance_dest	type_CASH_IN	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER
0	1	9839.64	0	-9839.64	0.0	0.0	0.0	0.0	1.0	0.0
1	1	1864.28	0	-1864.28	0.0	0.0	0.0	0.0	1.0	0.0
2	1	181.00	1	-181.00	0.0	0.0	0.0	0.0	0.0	1.0
3	1	181.00	1	-181.00	-21182.0	0.0	1.0	0.0	0.0	0.0
4	1	11668.14	0	-11668.14	0.0	0.0	0.0	0.0	1.0	0.0

In [221…

```python
df.corr()
```

Out[221]:

	step	amount	isFraud	dif_balance_org	dif_balance_dest	type_CASH_IN	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER
step	1.000000	-0.025996	0.045030	-0.010708	-0.083239	-0.005376	-0.013746	-0.005992	0.017102	0.004375
amount	-0.025996	1.000000	0.128862	-0.131801	0.513277	0.022341	0.071255	-0.047878	-0.397464	0.539278
isFraud	0.045030	0.128862	1.000000	-0.293467	0.032034	-0.017363	0.010328	-0.002741	-0.023566	0.049279
dif_balance_org	-0.010708	-0.131801	-0.293467	1.000000	-0.256157	0.582378	-0.277061	-0.014396	-0.135864	-0.151671
dif_balance_dest	-0.083239	0.513277	0.032034	-0.256157	1.000000	-0.216522	0.174827	-0.011314	-0.192612	0.353836
type_CASH_IN	-0.005376	0.022341	-0.017363	0.582378	-0.216522	1.000000	-0.391241	-0.043656	-0.375295	-0.157922
type_CASH_OUT	-0.013746	0.071255	0.010328	-0.277061	0.174827	-0.391241	1.000000	-0.061772	-0.531032	-0.223456
type_DEBIT	-0.005992	-0.047878	-0.002741	-0.014396	-0.011314	-0.043656	-0.061772	1.000000	-0.059254	-0.024934
type_PAYMENT	0.017102	-0.397464	-0.023566	-0.135864	-0.192612	-0.375295	-0.531032	-0.059254	1.000000	-0.214348
type_TRANSFER	0.004375	0.539278	0.049279	-0.151671	0.353836	-0.157922	-0.223456	-0.024934	-0.214348	1.000000

In [222…

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

Out[222]:

```
step                0
amount              0
isFraud             0
dif_balance_org     0
dif_balance_dest    0
type_CASH_IN        0
type_CASH_OUT       0
type_DEBIT          0
type_PAYMENT        0
type_TRANSFER       0
dtype: int64
```

In [223…

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 10 columns):
 #   Column            Non-Null Count    Dtype  
---  ------            --------------    -----  
 0   step              1048575 non-null  int64  
 1   amount            1048575 non-null  float64
 2   isFraud           1048575 non-null  int64  
 3   dif_balance_org   1048575 non-null  float64
 4   dif_balance_dest  1048575 non-null  float64
 5   type_CASH_IN      1048575 non-null  float64
 6   type_CASH_OUT     1048575 non-null  float64
 7   type_DEBIT        1048575 non-null  float64
 8   type_PAYMENT      1048575 non-null  float64
 9   type_TRANSFER     1048575 non-null  float64
dtypes: float64(8), int64(2)
memory usage: 80.0 MB
```

In [224…

```python
df['type_CASH_IN'] = df['type_CASH_IN'].astype(int)
df['type_CASH_OUT'] = df['type_CASH_OUT'].astype(int)
df['type_DEBIT'] = df['type_DEBIT'].astype(int)
df['type_PAYMENT'] = df['type_PAYMENT'].astype(int)
df['type_TRANSFER'] = df['type_TRANSFER'].astype(int)
```

In [225…

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 10 columns):
 #   Column            Non-Null Count    Dtype  
---  ------            --------------    -----  
 0   step              1048575 non-null  int64  
 1   amount            1048575 non-null  float64
 2   isFraud           1048575 non-null  int64  
 3   dif_balance_org   1048575 non-null  float64
 4   dif_balance_dest  1048575 non-null  float64
 5   type_CASH_IN      1048575 non-null  int32  
 6   type_CASH_OUT     1048575 non-null  int32  
 7   type_DEBIT        1048575 non-null  int32  
 8   type_PAYMENT      1048575 non-null  int32  
 9   type_TRANSFER     1048575 non-null  int32  
dtypes: float64(3), int32(5), int64(2)
memory usage: 60.0 MB
```

In [226…

```python
df.corr()
```

Out[226]:

	step	amount	isFraud	dif_balance_org	dif_balance_dest	type_CASH_IN	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER
step	1.000000	-0.025996	0.045030	-0.010708	-0.083239	-0.005376	-0.013746	-0.005992	0.017102	0.004375
amount	-0.025996	1.000000	0.128862	-0.131801	0.513277	0.022341	0.071255	-0.047878	-0.397464	0.539278
isFraud	0.045030	0.128862	1.000000	-0.293467	0.032034	-0.017363	0.010328	-0.002741	-0.023566	0.049279
dif_balance_org	-0.010708	-0.131801	-0.293467	1.000000	-0.256157	0.582378	-0.277061	-0.014396	-0.135864	-0.151671
dif_balance_dest	-0.083239	0.513277	0.032034	-0.256157	1.000000	-0.216522	0.174827	-0.011314	-0.192612	0.353836
type_CASH_IN	-0.005376	0.022341	-0.017363	0.582378	-0.216522	1.000000	-0.391241	-0.043656	-0.375295	-0.157922
type_CASH_OUT	-0.013746	0.071255	0.010328	-0.277061	0.174827	-0.391241	1.000000	-0.061772	-0.531032	-0.223456
type_DEBIT	-0.005992	-0.047878	-0.002741	-0.014396	-0.011314	-0.043656	-0.061772	1.000000	-0.059254	-0.024934
type_PAYMENT	0.017102	-0.397464	-0.023566	-0.135864	-0.192612	-0.375295	-0.531032	-0.059254	1.000000	-0.214348
type_TRANSFER	0.004375	0.539278	0.049279	-0.151671	0.353836	-0.157922	-0.223456	-0.024934	-0.214348	1.000000

In [227…

```python
 df["isFraud"].value_counts()
```

Out[227]:

```
0    1047433
1       1142
Name: isFraud, dtype: int64
```

test model of imbalanced dataset

In [228…

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,confusion_matrix,classification_report


# Split the data into features (X) and the target variable (y)
X = df.drop('isFraud', axis=1)  # Assuming 'isFraud' is the target variable
y = df['isFraud']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
rf_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"ROC AUC Score: {roc_auc}")
```

```
Accuracy: 0.999737739312877
Precision: 0.9378238341968912
Recall: 0.8080357142857143
F1 Score: 0.86810551558753
ROC AUC Score: 0.9039892162943244
```

oversampling

In [237…

```python
from imblearn.over_sampling import RandomOverSampler
```

In [ ]:

```python
os = RandomOverSampler()

X_trainos,y_trainos = os.fit_resample(X_train,y_train)

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
rf_classifier.fit(X_trainos, y_trainos)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"ROC AUC Score: {roc_auc}")
```

Smote

In [ ]:

```python
from imblearn.combine import SMOTETomek
os = SMOTETomek()
X_trainos,y_trainos = os.fit_resample(X_train,y_train)

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model on the training data
rf_classifier.fit(X_trainos, y_trainos)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"ROC AUC Score: {roc_auc}")
```

In [ ]: