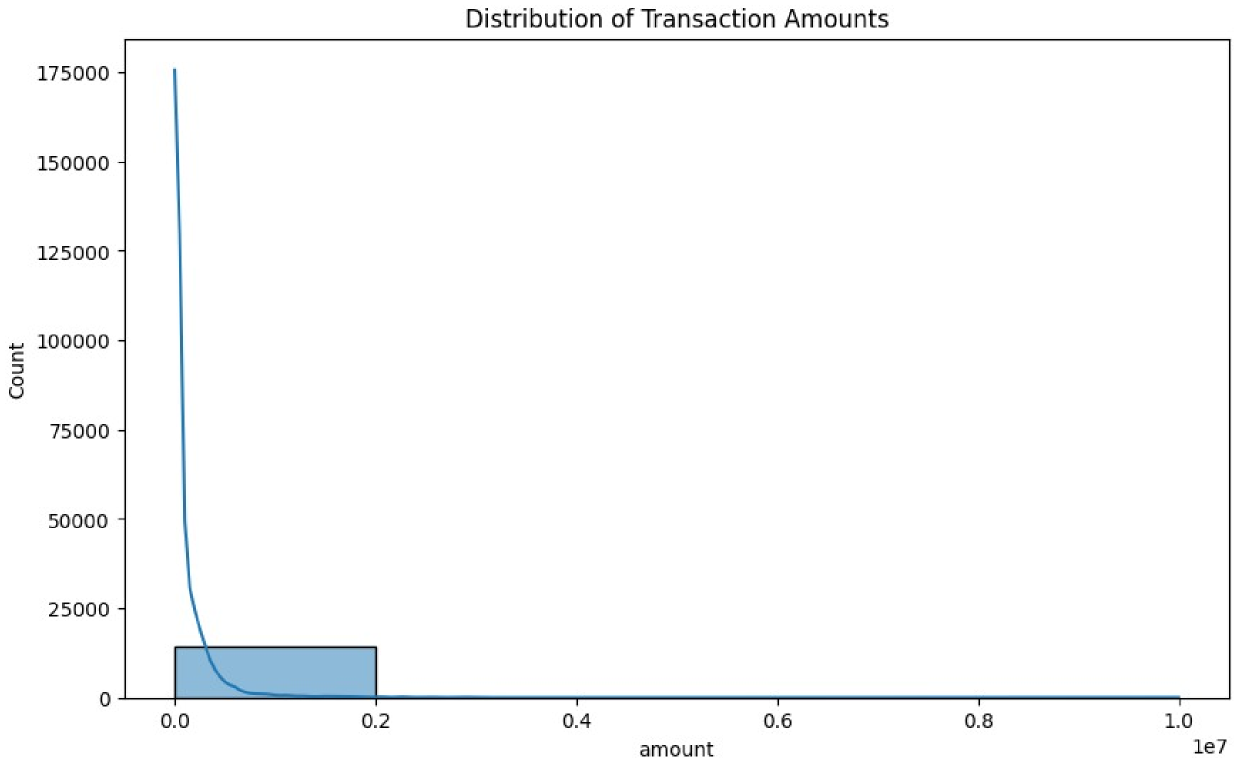
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB

data = pd.read_csv("/content/PS_20174392719_1491204439457_log.csv")
data.head()
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 14247,\n
\"fields\": [\n    {\n      \"column\": \"step\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2,\n        \"min\": 1,\n        \"max\": 8,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          2,\n
6,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"type\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"TRANSFER\",\n          \"CASH_IN\",\n          \"CASH_OUT\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"amount\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 280515.161534486,\
n        \"min\": 2.39,\n        \"max\": 10000000.0,\n
\"num_unique_values\": 14185,\n        \"samples\": [\n
14199.19,\n          7509.75,\n          273536.21\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"nameOrig\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
14247,\n        \"samples\": [\n          \"C1401212518\",\n
\"C636205886\",\n          \"C1272753974\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldbalanceOrg\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2016692.8588668816,\n        \"min\": 0.0,\n        \"max\":
12930418.44,\n        \"num_unique_values\": 10099,\n
\"samples\": [\n          271352.67,\n          1737.64,\n
1995.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"newbalanceOrig\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2059983.8008996155,\n        \"min\":
0.0,\n        \"max\": 13010502.78,\n        \"num_unique_values\":
8127,\n        \"samples\": [\n          96350.85,\n
12425.38,\n          18590.65\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"nameDest\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 9472,\n
\"samples\": [\n          \"M248487859\",\n          \"M1249676471\",\
n          \"M294485518\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"oldbalanceDest\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2528173.977265341,\n
\"min\": 0.0,\n        \"max\": 20937587.49,\n
\"num_unique_values\": 6081,\n        \"samples\": [\n
438038.9,\n          662260.23,\n          702449.21\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"newbalanceDest\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3151705.5251910673,\n        \"min\": 0.0,\n        \"max\":
25330272.63,\n        \"num_unique_values\": 2244,\n
\"samples\": [\n          1210658.29,\n          2226811.03,\n
158574.02\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFraud\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0.07379448031451728,\n        \"min\": 0.0,\n
\"max\": 1.0,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1.0,\n          0.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"isFlaggedFraud\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"data"}
```

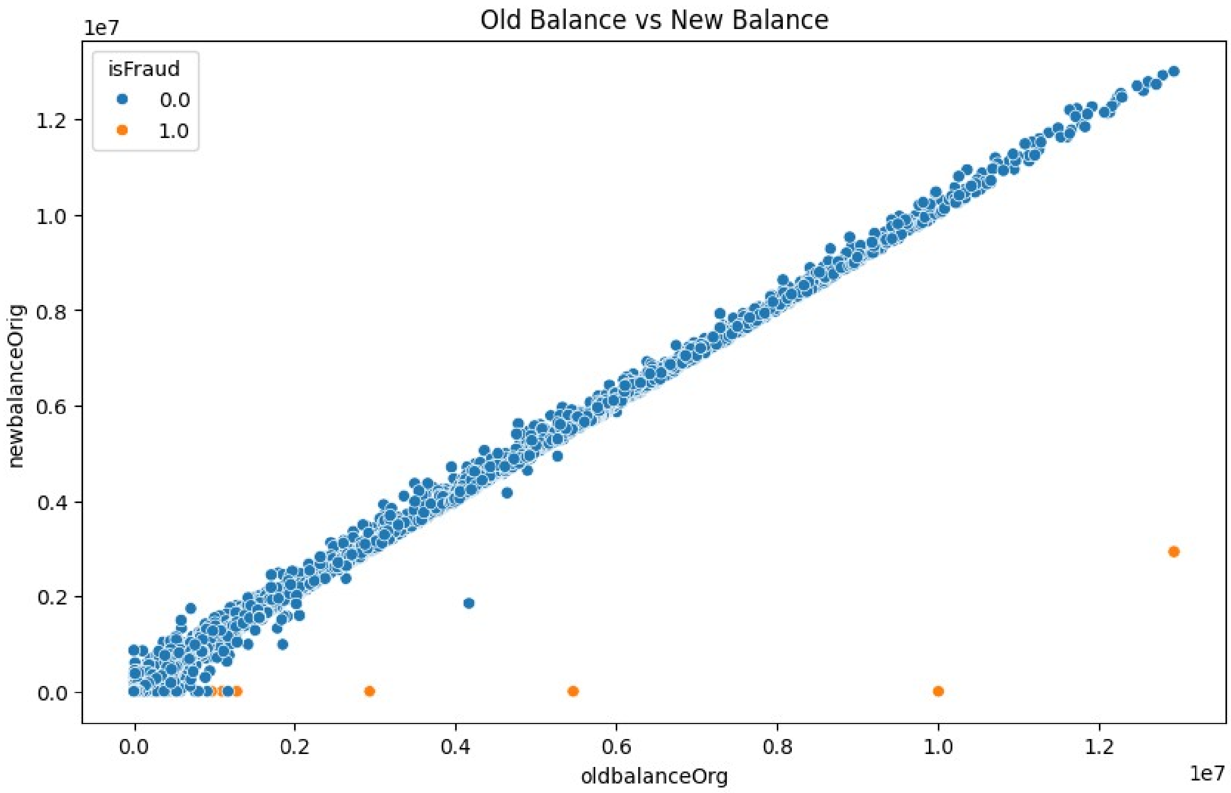
```
data.isnull().sum()
data = data.dropna()
data.isnull().sum()
data =
data[['amount','oldbalanceOrg','newbalanceOrig','oldbalanceDest','newb
alanceDest',"isFraud","isFlaggedFraud"]]
data.head()
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 14246,\n
\"fields\": [\n    {\n      \"column\": \"amount\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
280524.79986164556,\n        \"min\": 2.39,\n        \"max\":
10000000.0,\n        \"num_unique_values\": 14184,\n
\"samples\": [\n          14199.19,\n          7509.75,\n
154592.13\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"oldbalanceOrg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2016692.8588668816,\n        \"min\":
0.0,\n        \"max\": 12930418.44,\n        \"num_unique_values\":
10099,\n        \"samples\": [\n          271352.67,\n
1737.64,\n          1995.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"newbalanceOrig\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2059983.8008996155,\n
\"min\": 0.0,\n        \"max\": 13010502.78,\n
\"num_unique_values\": 8127,\n        \"samples\": [\n
96350.85,\n          12425.38,\n          18590.65\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldbalanceDest\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2528173.977265341,\n        \"min\": 0.0,\n        \"max\":
20937587.49,\n        \"num_unique_values\": 6081,\n
\"samples\": [\n          438038.9,\n          662260.23,\n
702449.21\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"newbalanceDest\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3151705.5251910673,\n        \"min\":
0.0,\n        \"max\": 25330272.63,\n        \"num_unique_values\":
2244,\n        \"samples\": [\n          1210658.29,\n
2226811.03,\n          158574.02\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"isFraud\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.07379448031451728,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFlaggedFraud\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.0,\n        \"min\": 0.0,\n
\"max\": 0.0,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```

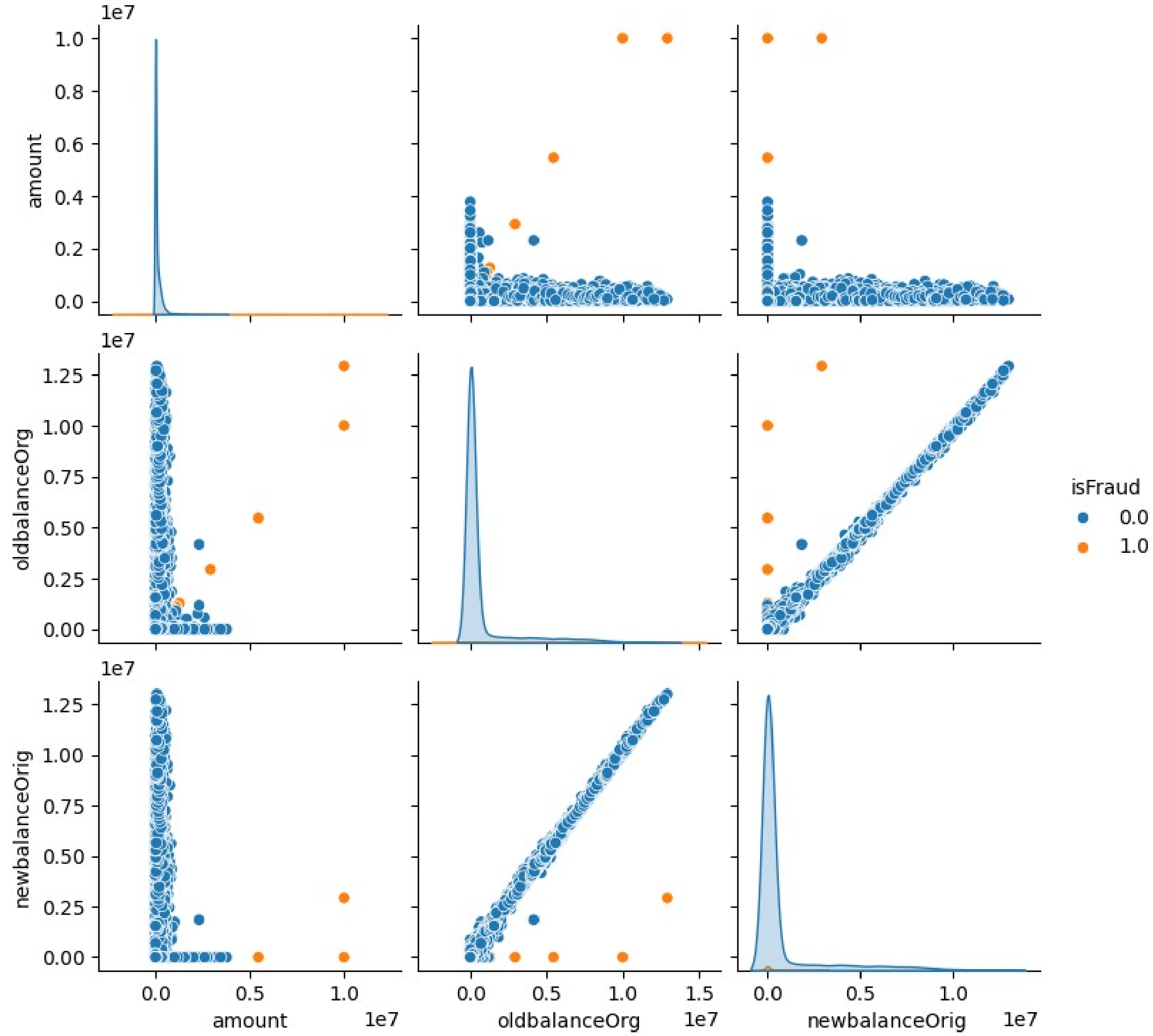
```python
plt.figure(figsize=(10, 6))
sns.histplot(data['amount'], bins=5, kde=True)
plt.title('Distribution of Transaction Amounts')
plt.show()
```



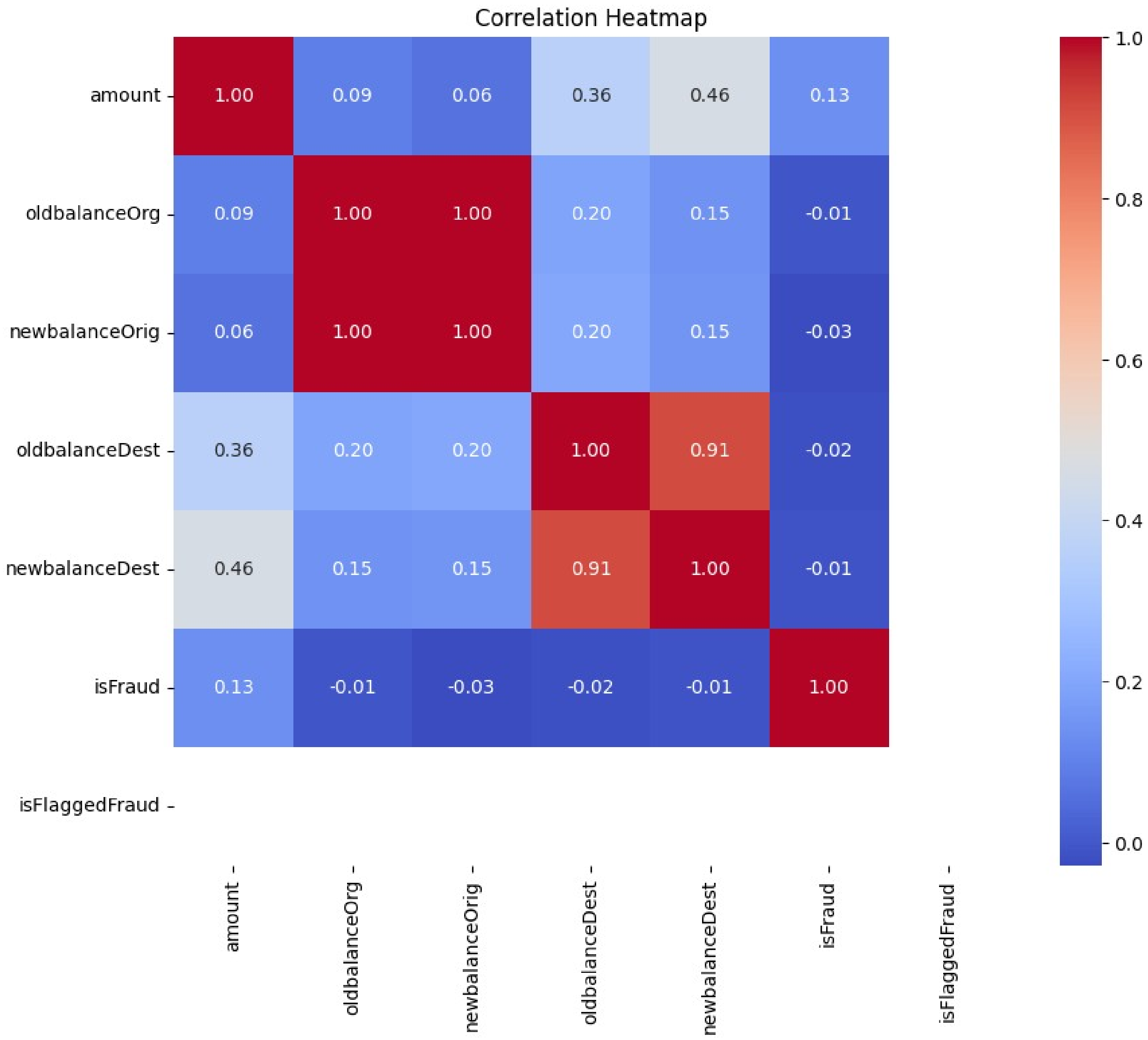
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='oldbalanceOrg', y='newbalanceOrig', hue='isFraud',
data=data)
plt.title('Old Balance vs New Balance')
plt.show()
```



```
print(data.columns)

Index(['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
       'newbalanceDest', 'isFraud', 'isFlaggedFraud'],
      dtype='object')

sns.pairplot(data[['amount', 'oldbalanceOrg', 'newbalanceOrig',
'isFraud']], hue='isFraud')
plt.show()
```



```
corr_matrix = data.corr()

# Generate a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



```
data['isFraud'].value_counts()
data.head()
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 14246,\n
\"fields\": [\n    {\n      \"column\": \"amount\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
280524.79986164556,\n        \"min\": 2.39,\n        \"max\":
10000000.0,\n        \"num_unique_values\": 14184,\n
\"samples\": [\n          14199.19,\n          7509.75,\n
154592.13\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"oldbalanceOrg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2016692.8588668816,\n        \"min\":
0.0,\n        \"max\": 12930418.44,\n        \"num_unique_values\":
10099,\n        \"samples\": [\n          271352.67,\n
1737.64,\n          1995.0\n        ],\n        \"semantic_type\":
```

```
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"newbalanceOrig\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2059983.8008996155,\n
\"min\": 0.0,\n        \"max\": 13010502.78,\n
\"num_unique_values\": 8127,\n        \"samples\": [\n
96350.85,\n          12425.38,\n          18590.65\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldbalanceDest\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2528173.977265341,\n        \"min\": 0.0,\n        \"max\":
20937587.49,\n        \"num_unique_values\": 6081,\n
\"samples\": [\n          438038.9,\n          662260.23,\n
702449.21\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"newbalanceDest\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 3151705.5251910673,\n        \"min\":
0.0,\n        \"max\": 25330272.63,\n        \"num_unique_values\":
2244,\n        \"samples\": [\n          1210658.29,\n
2226811.03,\n          158574.02\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"isFraud\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.07379448031451728,\n        \"min\": 0.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1.0,\n
0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"isFlaggedFraud\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.0,\n        \"min\": 0.0,\n
\"max\": 0.0,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          0.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

x = data.drop(['isFraud'],axis = 1)
y = data['isFraud']

x_scaled = scaler.fit_transform(x)
print(x_scaled)
```

```
[[-0.3637883  -0.30923044 -0.31575007 -0.33271566 -0.36414239  0.
]
 [-0.39221944 -0.38306034 -0.38415672 -0.33271566 -0.36414239  0.
]
 [-0.39822011 -0.39350751 -0.39356718 -0.33271566 -0.36414239  0.
]
 ...
 [-0.13062818 -0.3745709  -0.39356718 -0.3326496  -0.36414239  0.
]
 [-0.38790566 -0.38851848 -0.39008759 -0.33271566 -0.36414239  0.
]
 [-0.35799335 -0.39357842 -0.39356718 -0.33271566 -0.36414239  0.
]]
```

```python
print(x)
print(y)
```

```
         amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
0       9839.64       170136.0       160296.36             0.0
1       1864.28        21249.0        19384.72             0.0
2        181.00          181.0            0.00             0.0
3        181.00          181.0            0.00         21182.0
4      11668.14        41554.0        29885.86             0.0
...         ...            ...             ...             ...
14241  35108.12            0.0            0.00             0.0
14242  20924.47        18265.0            0.00             0.0
14243  75244.54        38369.0            0.00           167.0
14244   3074.36        10242.0         7167.64             0.0
14245  11465.21           38.0            0.00             0.0

       newbalanceDest  isFlaggedFraud
0                 0.0             0.0
1                 0.0             0.0
2                 0.0             0.0
3                 0.0             0.0
4                 0.0             0.0
...               ...             ...
14241             0.0             0.0
14242             0.0             0.0
14243             0.0             0.0
14244             0.0             0.0
14245             0.0             0.0

[14246 rows x 6 columns]
0        0.0
1        0.0
2        1.0
3        1.0
4        0.0
        ...
14241    0.0
14242    0.0
14243    0.0
14244    0.0
14245    0.0
Name: isFraud, Length: 14246, dtype: float64
```

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test =
train_test_split(x_scaled,y,test_size=0.20, random_state=42)

model = GaussianNB()
model.fit(x_train,y_train)
```

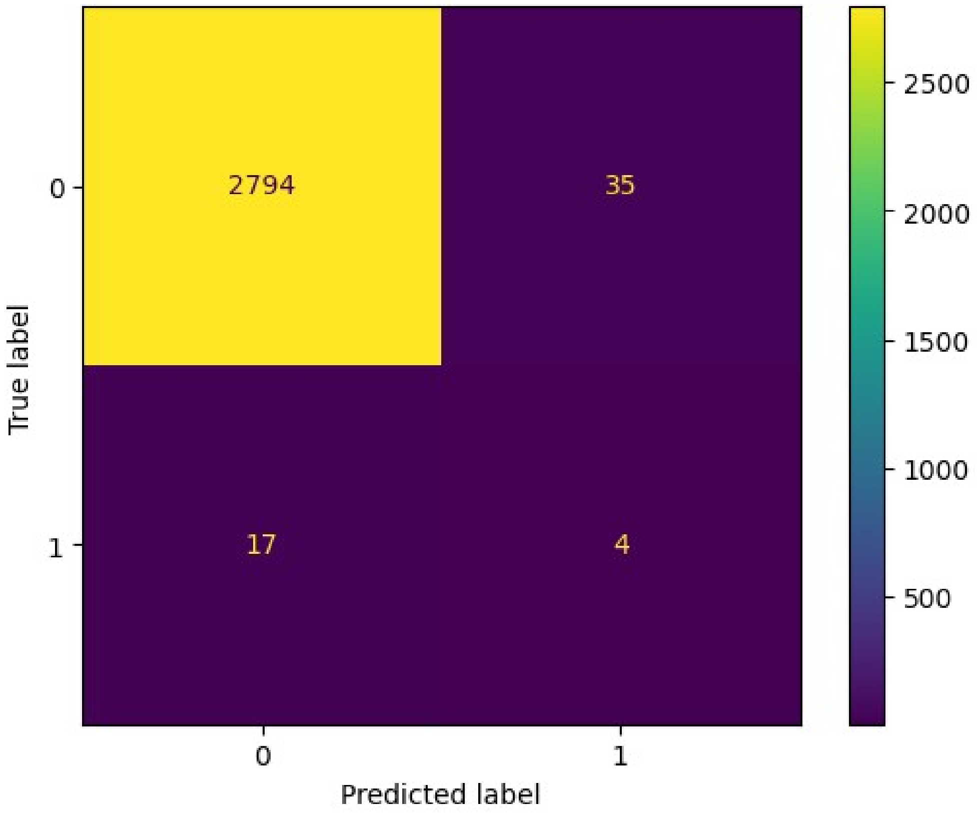
```
GaussianNB()
```

```
predict = model.predict(x_test)
predict
```

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

```
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
)

y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred, average="weighted")

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()

print("Accuracy for Naive Bayes:", accuracy)
print("F1 Score for Naive Bayes:", f1)
```

```
Accuracy for Naive Bayes: 0.9817543859649123
F1 Score for Naive Bayes: 0.9844621127286302
```



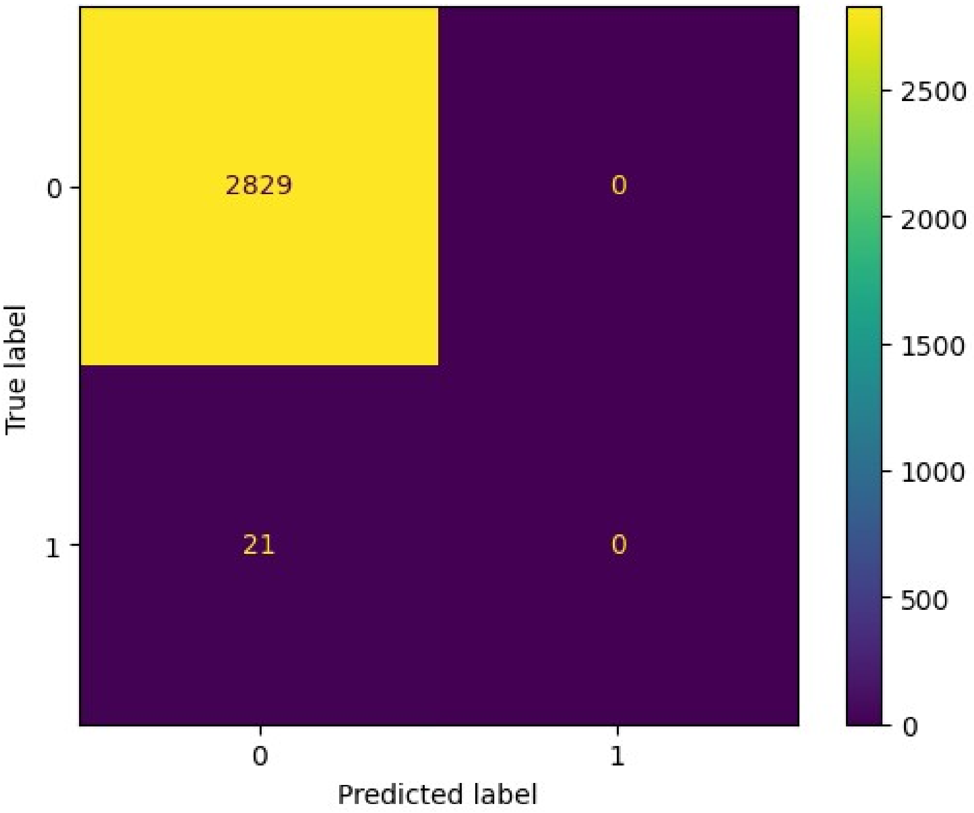
```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
clf = SVC(kernel='rbf', random_state=42)

clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)

# Evaluate the classifier
print("Accuracy for SVM :", accuracy_score(y_test, y_pred))
print("Confusion Matrix for SVM:\n", confusion_matrix(y_test, y_pred))
print("Classification Report of SVM :\n",
classification_report(y_test, y_pred))
```

```
Accuracy for SVM : 0.9926315789473684
Confusion Matrix for SVM:
 [[2829    0]
 [  21    0]]
Classification Report of SVM :
               precision    recall  f1-score   support

         0.0       0.99      1.00      1.00      2829
         1.0       0.00      0.00      0.00        21
```

```
    accuracy                           0.99      2850
   macro avg       0.50      0.50      0.50      2850
weighted avg       0.99      0.99      0.99      2850


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
cm =  confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7e66ab85e290>
```


0
1
True label
2829
0
21
0
0
1
Predicted label
2500
2000
1500
1000
500
0