In [1]:

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


pd.set_option('display.max_columns', 80)
```

In [2]:

```python
df = pd.read_csv('zindi training.csv')
df.head(5)
```

Out[2]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	Cu
0	TransactionId_76871	BatchId_36123	AccountId_3957	SubscriptionId_887	CustomerId_4406	
1	TransactionId_73770	BatchId_15642	AccountId_4841	SubscriptionId_3829	CustomerId_4406	
2	TransactionId_26203	BatchId_53941	AccountId_4229	SubscriptionId_222	CustomerId_4683	
3	TransactionId_380	BatchId_102363	AccountId_648	SubscriptionId_2185	CustomerId_988	
4	TransactionId_28195	BatchId_38780	AccountId_4841	SubscriptionId_3829	CustomerId_988	

In [3]:

```python
df.shape
```

Out[3]:

```
(95662, 16)
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95662 entries, 0 to 95661
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   TransactionId         95662 non-null  object 
 1   BatchId               95662 non-null  object 
 2   AccountId             95662 non-null  object 
 3   SubscriptionId        95662 non-null  object 
 4   CustomerId            95662 non-null  object 
 5   CurrencyCode          95662 non-null  object 
 6   CountryCode           95662 non-null  int64  
 7   ProviderId            95662 non-null  object 
 8   ProductId             95662 non-null  object 
 9   ProductCategory       95662 non-null  object 
 10  ChannelId             95662 non-null  object 
 11  Amount                95662 non-null  float64
 12  Value                 95662 non-null  int64  
 13  TransactionStartTime  95662 non-null  object 
 14  PricingStrategy       95662 non-null  int64  
 15  FraudResult           95662 non-null  int64  
dtypes: float64(1), int64(4), object(11)
memory usage: 11.7+ MB
```

In [5]:

```
categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variabes \n'.format(len(categorical)))

print('They are: ', categorical)
```

```
There are 11 categorical variabes 

They are:  ['TransactionId', 'BatchId', 'AccountId', 'SubscriptionId', 'Cust
omerId', 'CurrencyCode', 'ProviderId', 'ProductId', 'ProductCategory', 'Chan
nelId', 'TransactionStartTime']
```

In [6]:

```
df[categorical].head()
```

Out[6]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	Cu
0	TransactionId_76871	BatchId_36123	AccountId_3957	SubscriptionId_887	CustomerId_4406	
1	TransactionId_73770	BatchId_15642	AccountId_4841	SubscriptionId_3829	CustomerId_4406	
2	TransactionId_26203	BatchId_53941	AccountId_4229	SubscriptionId_222	CustomerId_4683	
3	TransactionId_380	BatchId_102363	AccountId_648	SubscriptionId_2185	CustomerId_988	
4	TransactionId_28195	BatchId_38780	AccountId_4841	SubscriptionId_3829	CustomerId_988	

In [7]:

```
df[categorical].isnull().sum()
```

Out[7]:

```
TransactionId           0
BatchId                 0
AccountId               0
SubscriptionId          0
CustomerId              0
CurrencyCode            0
ProviderId              0
ProductId               0
ProductCategory         0
ChannelId               0
TransactionStartTime    0
dtype: int64
```

In [8]:

```
for var in categorical:
    print(var, 'contains', len(df[var].unique()), 'unique values')
```

```
TransactionId contains 95662 unique values
BatchId contains 94809 unique values
AccountId contains 3633 unique values
SubscriptionId contains 3627 unique values
CustomerId contains 3742 unique values
CurrencyCode contains 1 unique values
ProviderId contains 6 unique values
ProductId contains 23 unique values
ProductCategory contains 9 unique values
ChannelId contains 4 unique values
TransactionStartTime contains 94556 unique values
```

In [9]:

```
df['TransactionStartTime'] = pd.to_datetime(df['TransactionStartTime'])
```

In [10]:

```
df['Year'] = df['TransactionStartTime'].dt.year
df['Year'].head()
```

Out[10]:

```
0    2018
1    2018
2    2018
3    2018
4    2018
Name: Year, dtype: int64
```

In [11]:

```
df['Month'] = df['TransactionStartTime'].dt.month
df['Month'].head()
```

Out[11]:

```
0    11
1    11
2    11
3    11
4    11
Name: Month, dtype: int64
```

In [12]:

```
#extract the day


df['day'] = df['TransactionStartTime'].dt.day
df['day'].head()
```

Out[12]:

```
0    15
1    15
2    15
3    15
4    15
Name: day, dtype: int64
```

In [13]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95662 entries, 0 to 95661
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype              
---  ------                --------------  -----              
 0   TransactionId         95662 non-null  object             
 1   BatchId               95662 non-null  object             
 2   AccountId             95662 non-null  object             
 3   SubscriptionId        95662 non-null  object             
 4   CustomerId            95662 non-null  object             
 5   CurrencyCode          95662 non-null  object             
 6   CountryCode           95662 non-null  int64              
 7   ProviderId            95662 non-null  object             
 8   ProductId             95662 non-null  object             
 9   ProductCategory       95662 non-null  object             
 10  ChannelId             95662 non-null  object             
 11  Amount                95662 non-null  float64            
 12  Value                 95662 non-null  int64              
 13  TransactionStartTime  95662 non-null  datetime64[ns, UTC]
 14  PricingStrategy       95662 non-null  int64              
 15  FraudResult           95662 non-null  int64              
 16  Year                  95662 non-null  int64              
 17  Month                 95662 non-null  int64              
 18  day                   95662 non-null  int64              
dtypes: datetime64[ns, UTC](1), float64(1), int64(7), object(10)
memory usage: 13.9+ MB
```

In [14]:

```
df.drop('TransactionStartTime',axis = 1, inplace=True)
df
```

Out[14]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId
0	TransactionId_76871	BatchId_36123	AccountId_3957	SubscriptionId_887	CustomerId_4406
1	TransactionId_73770	BatchId_15642	AccountId_4841	SubscriptionId_3829	CustomerId_4406
2	TransactionId_26203	BatchId_53941	AccountId_4229	SubscriptionId_222	CustomerId_4683
3	TransactionId_380	BatchId_102363	AccountId_648	SubscriptionId_2185	CustomerId_988
4	TransactionId_28195	BatchId_38780	AccountId_4841	SubscriptionId_3829	CustomerId_988
...	...	...	...	...	..
95657	TransactionId_89881	BatchId_96668	AccountId_4841	SubscriptionId_3829	CustomerId_3078
95658	TransactionId_91597	BatchId_3503	AccountId_3439	SubscriptionId_2643	CustomerId_3874
95659	TransactionId_82501	BatchId_118602	AccountId_4841	SubscriptionId_3829	CustomerId_3874
95660	TransactionId_136354	BatchId_70924	AccountId_1346	SubscriptionId_652	CustomerId_1709
95661	TransactionId_35670	BatchId_29317	AccountId_4841	SubscriptionId_3829	CustomerId_1709

95662 rows × 18 columns

In [15]:

```
#we'll extraxt the categorical features again since we have modified our data
categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variabes \n'.format(len(categorical)))

print('They are: ', categorical)
```

```
There are 10 categorical variabes 

They are:  ['TransactionId', 'BatchId', 'AccountId', 'SubscriptionId', 'Cust
omerId', 'CurrencyCode', 'ProviderId', 'ProductId', 'ProductCategory', 'Chan
nelId']
```

In [16]:

```
df[categorical].isnull().sum()
```

Out[16]:

```
TransactionId      0
BatchId            0
AccountId          0
SubscriptionId     0
CustomerId         0
CurrencyCode       0
ProviderId         0
ProductId          0
ProductCategory    0
ChannelId          0
dtype: int64
```

In [17]:

```
len(df.TransactionId.unique())
```

Out[17]:

```
95662
```

In [18]:

```
#let's get dummies for this columns
providerId_dummy = pd.get_dummies(df.ProviderId, drop_first=True)
providerId_dummy
```

Out[18]:

	ProviderId_2	ProviderId_3	ProviderId_4	ProviderId_5	ProviderId_6
0	0	0	0	0	1
1	0	0	1	0	0
2	0	0	0	0	1
3	0	0	0	0	0
4	0	0	1	0	0
...	...	...	...	...	...
95657	0	0	1	0	0
95658	0	0	0	0	1
95659	0	0	1	0	0
95660	0	0	0	0	1
95661	0	0	1	0	0

95662 rows × 5 columns

In [19]:

```
productId_dummy = pd.get_dummies(df.ProductId, drop_first=True)
productId_dummy
```

Out[19]:

	ProductId_10	ProductId_11	ProductId_12	ProductId_13	ProductId_14	ProductId_15	Pr
0	1	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	
...	...	...	...	...	...	...	
95657	0	0	0	0	0	0	
95658	1	0	0	0	0	0	
95659	0	0	0	0	0	0	
95660	0	0	0	0	0	0	
95661	0	0	0	0	0	0	

95662 rows × 22 columns

In [20]:

```
ProductCategory_dummy = pd.get_dummies(df.ProductCategory, drop_first=True)
ProductCategory_dummy

```

Out[20]:

	data_bundles	financial_services	movies	other	ticket	transport	tv	utility_bill
0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1
4	0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
95657	0	1	0	0	0	0	0	0
95658	0	0	0	0	0	0	0	0
95659	0	1	0	0	0	0	0	0
95660	0	0	0	0	0	0	1	0
95661	0	1	0	0	0	0	0	0

95662 rows × 8 columns

In [21]:

```
ChannelId_dummy = pd.get_dummies(df.ChannelId, drop_first=True)
ChannelId_dummy

```

Out[21]:

	ChannelId_2	ChannelId_3	ChannelId_5
0	0	1	0
1	1	0	0
2	0	1	0
3	0	1	0
4	1	0	0
...	...	...	...
95657	1	0	0
95658	0	1	0
95659	1	0	0
95660	0	1	0
95661	1	0	0

95662 rows × 3 columns

In [22]:

```
PricingStrategy_dummy = pd.get_dummies(df.PricingStrategy, drop_first=True)
PricingStrategy_dummy

```

Out[22]:

	1	2	4
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	0	1	0
...	...	...	...
95657	0	1	0
95658	0	1	0
95659	0	1	0
95660	0	1	0
95661	0	1	0

95662 rows × 3 columns

In [23]:

```
#we'll extraxt the numerical features
numerical = [var for var in df.columns if df[var].dtype!='O']

print('There are {} numerical variabes \n'.format(len(numerical)))

print('They are: ', numerical)
```

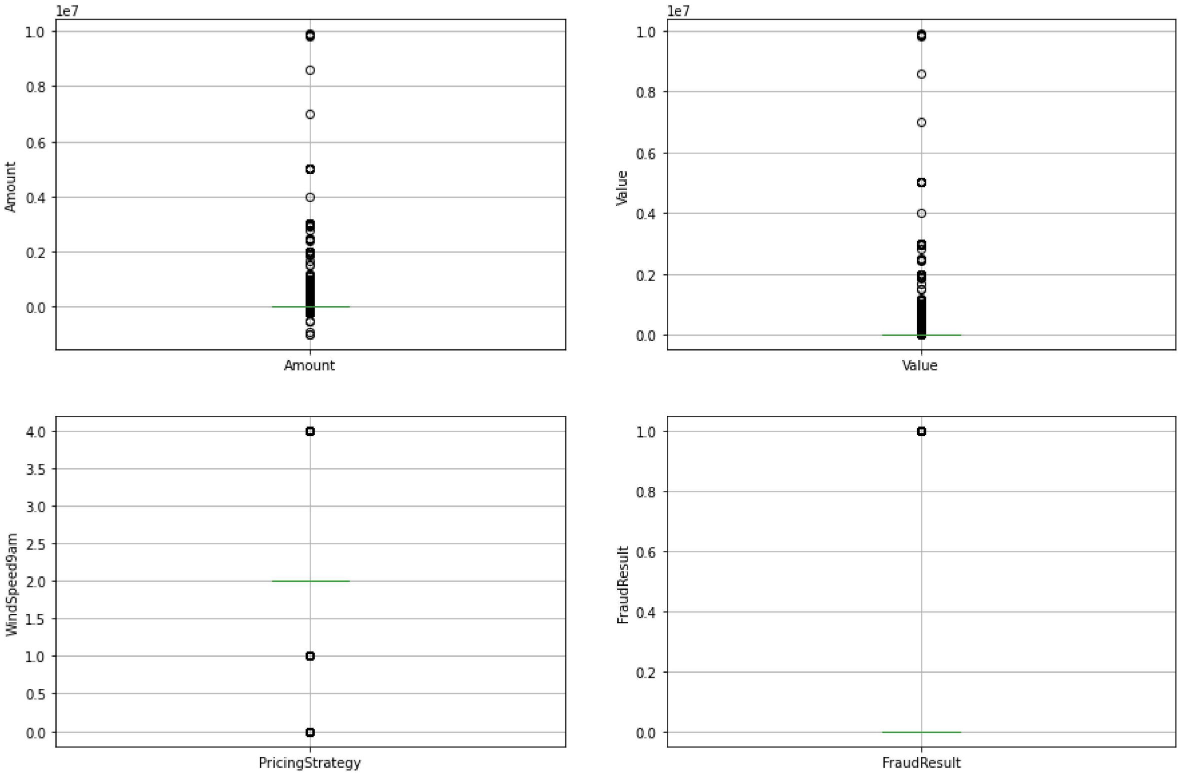
```
There are 8 numerical variabes 

They are:  ['CountryCode', 'Amount', 'Value', 'PricingStrategy', 'FraudResul
t', 'Year', 'Month', 'day']
```

In [24]:

```
plt.figure(figsize=(15,10))

plt.subplot(2,2,1)
fig = df.boxplot(column='Amount')
fig.set_ylabel('Amount')

plt.subplot(2,2,2)
fig = df.boxplot(column='Value')
fig.set_ylabel('Value')

plt.subplot(2,2,3)
fig = df.boxplot(column='PricingStrategy')
fig.set_ylabel('WindSpeed9am')

plt.subplot(2,2,4)
fig = df.boxplot(column='FraudResult')
fig.set_ylabel('FraudResult')
```

Out[24]:

```
Text(0, 0.5, 'FraudResult')
```

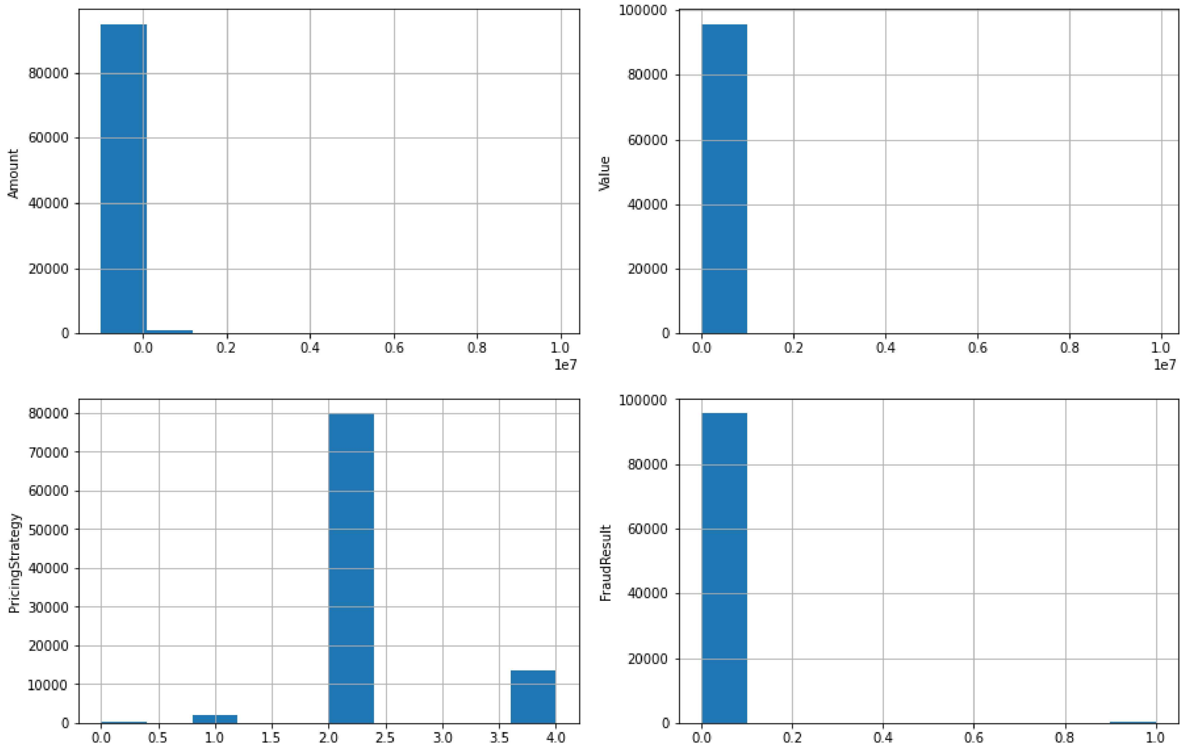


In [25]:

```
#set figure size
plt.figure(figsize=(15,10))

plt.subplot(2,2,1)
fig = df.Amount.hist()
fig.set_ylabel('Amount')

plt.subplot(2,2,2)
fig = df.Value.hist()
fig.set_ylabel('Value')


plt.subplot(2,2,3)
fig = df.PricingStrategy.hist()
fig.set_ylabel('PricingStrategy')

plt.subplot(2,2,4)
fig = df.FraudResult.hist()
fig.set_ylabel('FraudResult')
```

Out[25]:

```
Text(0, 0.5, 'FraudResult')
```



In [26]:

```
new_df = df['BatchId'].str.split('_',n=1,expand = True)

df['BatchId'] = new_df[1].astype(int)
```

In [27]:

```
new_df1 = df['CustomerId'].str.split('_',n=1,expand = True)

df['CustomerId'] = new_df[1].astype(int)
```

In [28]:

```
new_df = df['TransactionId'].str.split('_',n=1,expand = True)

df['TransactionId'] = new_df[1].astype(int)
```

In [29]:

```
new_df = df['AccountId'].str.split('_',n=1,expand = True)

df['AccountId'] = new_df[1].astype(int)
```

In [30]:

```
new_df = df['SubscriptionId'].str.split('_',n=1,expand = True)

df['SubscriptionId'] = new_df[1].astype(int)
```

In [31]:

```
new_df = df['ProviderId'].str.split('_',n=1,expand = True)

df['ProviderId'] = new_df[1].astype(int)
```

In [32]:

```
new_df = df['ProductId'].str.split('_',n=1,expand = True)

df['ProductId'] = new_df[1].astype(int)
```

In [33]:

```
new_df = df['ChannelId'].str.split('_',n=1,expand = True)

df['ChannelId'] = new_df[1].astype(int)
```

In [45]:

```
df
```

Out[45]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	CurrencyCode	CountryC
0	76871	36123	3957	887	36123	UGX	.
1	73770	15642	4841	3829	15642	UGX	.
2	26203	53941	4229	222	53941	UGX	.
3	380	102363	648	2185	102363	UGX	.
4	28195	38780	4841	3829	38780	UGX	.
...	...	...	...	...	...	...	
95657	89881	96668	4841	3829	96668	UGX	.
95658	91597	3503	3439	2643	3503	UGX	.
95659	82501	118602	4841	3829	118602	UGX	.
95660	136354	70924	1346	652	70924	UGX	.
95661	35670	29317	4841	3829	29317	UGX	.

95662 rows × 18 columns

In [47]:

```
df.drop(['CurrencyCode','CountryCode',], axis = 1, inplace = True)
df.head()
```

Out[47]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	ProviderId	ProductId	Produc
0	76871	36123	3957	887	36123	6	10	
1	73770	15642	4841	3829	15642	4	6	financi
2	26203	53941	4229	222	53941	6	1	
3	380	102363	648	2185	102363	1	21	
4	28195	38780	4841	3829	38780	4	6	financi

In [48]:

```
#getting dummies for some columns that are categorical
selected_categorical_columns = ["ProviderId", "ProductId", "ProductCategory","ChannelI
new_df = pd.get_dummies(df, columns= selected_categorical_columns, drop_first = True)
new_df.head()
```

Out[48]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	Amount	Value	FraudResult
0	76871	36123	3957	887	36123	1000.0	1000	0
1	73770	15642	4841	3829	15642	-20.0	20	0
2	26203	53941	4229	222	53941	500.0	500	0
3	380	102363	648	2185	102363	20000.0	21800	0
4	28195	38780	4841	3829	38780	-644.0	644	0

In [49]:

```
new_df.shape
```

Out[49]:

```
(95662, 52)
```

In [50]:

```
#split data into target and feature

X = new_df.drop(['FraudResult'], axis =1)
y= new_df['FraudResult']
```

In [72]:

```
#split into test and train data

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [73]:

```

X_train.shape, X_test.shape
```

Out[73]:

```
((76529, 51), (19133, 51))
```

In [74]:

```
X_train
```

Out[74]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	Amount	Value	Year	M
66339	140085	114864	4841	3829	114864	-50.0	50	2019	
87279	64558	39429	3981	910	39429	1000.0	1000	2019	
40582	134904	1673	135	3595	1673	15000.0	16650	2018	
58655	95030	133112	4840	3829	133112	-1000.0	1000	2019	
87335	75383	7649	4841	3829	7649	-50.0	50	2019	
...	...	...	...	...	...	...	...	...	
21243	1505	124507	575	4369	124507	2000.0	2000	2018	
45891	68325	94167	4841	3829	94167	-50.0	50	2018	
42613	62369	85767	571	873	85767	1500.0	1500	2018	
43567	1639	36597	2123	1456	36597	2000.0	2000	2018	
68268	83620	99997	2659	3327	99997	1000.0	1000	2019	

76529 rows × 51 columns

In [71]:

```
X_test
```

Out[71]:

```
array([[0.64961075, 0.84289658, 0.17210744, ..., 0.        , 0.        ,
        1.        ],
       [0.84885002, 0.17318516, 0.85247934, ..., 0.        , 1.        ,
        0.        ],
       [0.88152572, 0.36934935, 0.03904959, ..., 0.        , 1.        ,
        0.        ],
       ...,
       [0.96174327, 0.23050726, 0.6142562 , ..., 0.        , 1.        ,
        0.        ],
       [0.78433045, 0.4117297 , 0.54690083, ..., 0.        , 1.        ,
        0.        ],
       [0.75759838, 0.49365119, 0.50454545, ..., 0.        , 1.        ,
        0.        ]])
```

In [56]:

```
X_train.dtypes
```

Out[56]:

```
TransactionId                          int32
BatchId                                int32
AccountId                              int32
SubscriptionId                         int32
CustomerId                             int32
Amount                               float64
Value                                  int64
Year                                   int64
Month                                  int64
day                                    int64
ProviderId_2                           uint8
ProviderId_3                           uint8
ProviderId_4                           uint8
ProviderId_5                           uint8
ProviderId_6                           uint8
ProductId_2                            uint8
ProductId_3                            uint8
ProductId_4                            uint8
```

In [76]:

```
#feature scaling

from sklearn.preprocessing import MinMaxScaler
scaler= MinMaxScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [63]:

```
X_train
```

Out[63]:

```
array([[0.99576343, 0.82343664, 1.        , ..., 0.        , 1.        ,
        0.        ],
       [0.45889252, 0.28264906, 0.82231405, ..., 0.        , 0.        ,
        1.        ],
       [0.95893517, 0.01197927, 0.02768595, ..., 0.        , 1.        ,
        0.        ],
       ...,
       [0.44333239, 0.61484253, 0.1177686 , ..., 0.        , 1.        ,
        0.        ],
       [0.01164345, 0.26234667, 0.43842975, ..., 0.        , 1.        ,
        0.        ],
       [0.59439153, 0.71685628, 0.54917355, ..., 0.        , 1.        ,
        0.        ]])
```

In [64]:

```
X_test
```

Out[64]:

```
array([[0.64961075, 0.84289658, 0.17210744, ..., 0.        , 0.        ,
        1.        ],
       [0.84885002, 0.17318516, 0.85247934, ..., 0.        , 1.        ,
        0.        ],
       [0.88152572, 0.36934935, 0.03904959, ..., 0.        , 1.        ,
        0.        ],
       ...,
       [0.96174327, 0.23050726, 0.6142562 , ..., 0.        , 1.        ,
        0.        ],
       [0.78433045, 0.4117297 , 0.54690083, ..., 0.        , 1.        ,
        0.        ],
       [0.75759838, 0.49365119, 0.50454545, ..., 0.        , 1.        ,
        0.        ]])
```

In [75]:

```
cols = X_train.columns
```

In [79]:

```
X_test = pd.DataFrame(X_test, columns=[cols])
X_test
```

Out[79]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	Amount	Value	Yea
0	0.649611	0.842897	0.172107	0.777203	0.842897	0.092258	0.000203	1.
1	0.848850	0.173185	0.852479	0.876707	0.173185	0.092176	0.000112	1.
2	0.881526	0.369349	0.039050	0.392015	0.369349	0.092258	0.000203	0.
3	0.086950	0.909783	0.999793	0.791684	0.909783	0.091981	0.000101	1.
4	0.192229	0.127887	0.326033	0.169425	0.127887	0.092166	0.000101	1.
...	...	...	...	...	...	...	...	.
19128	0.948576	0.287148	0.820248	0.320025	0.287148	0.092258	0.000203	1.
19129	0.548726	0.905682	1.000000	0.791684	0.905682	0.091613	0.000507	1.
19130	0.961743	0.230507	0.614256	0.476210	0.230507	0.092248	0.000192	1.
19131	0.784330	0.411730	0.546901	0.270790	0.411730	0.092166	0.000101	1.
19132	0.757598	0.493651	0.504545	0.729003	0.493651	0.092258	0.000203	0.

19133 rows × 51 columns

In [ ]:

```

```

In [78]:

```
X_train = pd.DataFrame(X_train, columns=[cols])
X_train
```

Out[78]:

	TransactionId	BatchId	AccountId	SubscriptionId	CustomerId	Amount	Value	Yea
0	0.995763	0.823437	1.000000	0.791684	0.823437	0.083483	0.000005	1.
1	0.458893	0.282649	0.822314	0.187836	0.282649	0.083581	0.000101	1.
2	0.958935	0.011979	0.027686	0.743277	0.011979	0.084879	0.001685	0.
3	0.675498	0.954255	0.999793	0.791684	0.954255	0.083395	0.000101	1.
4	0.535840	0.054821	1.000000	0.791684	0.054821	0.083483	0.000005	1.
...	...	...	...	...	...	...	...	.
76524	0.010691	0.892567	0.118595	0.903393	0.892567	0.083673	0.000202	0.
76525	0.485670	0.675061	1.000000	0.791684	0.675061	0.083483	0.000005	0.
76526	0.443332	0.614843	0.117769	0.180182	0.614843	0.083627	0.000152	0.
76527	0.011643	0.262347	0.438430	0.300786	0.262347	0.083673	0.000202	0.
76528	0.594392	0.716856	0.549174	0.687836	0.716856	0.083581	0.000101	1.

76529 rows × 51 columns

In [80]:

```
#Model training
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(solver='liblinear', random_state=0)

logreg.fit(X_train, y_train)
```

Out[80]:

```
LogisticRegression(random_state=0, solver='liblinear')
```

In [81]:

```
#test data
y_pred_test = logreg.predict(X_test)
```

In [82]:

```
y_pred_test
```

Out[82]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [83]:

```
#probability of getting zero, i.e , no fraud

logreg.predict_proba(X_test)[:,0]
```

Out[83]:

```
array([0.99753105, 0.99977752, 0.99987278, ..., 0.99418927, 0.99983678,
       0.9998944 ])
```

In [84]:

```
#check for accuracy

from sklearn.metrics import accuracy_score

print('Model accuracy score(test): ', accuracy_score(y_test, y_pred_test))
```

```
Model accuracy score(test):  0.998484294151466
```

In [85]:

```
y_pred_train = logreg.predict(X_train)

print('Model accuracy score(train): ', accuracy_score(y_train, y_pred_train))
```

```
Model accuracy score(train):  0.99820982895373
```

In [86]:

```
#model accuracy metrics for logistic regression
print('Training set score: ', logreg.score(X_train, y_train))

print('Test set score: ', logreg.score(X_test, y_test))
```

```
Training set score:  0.99820982895373
Test set score:  0.998484294151466
```

In [87]:

```
#check null accuracy
null_acuracy = 22726/(len(y_test))
```

In [88]:

```
null_acuracy
```

Out[88]:

```
1.1877907280614646
```

In [89]:

```
from sklearn.metrics import confusion_matrix
cm =confusion_matrix(y_test, y_pred_test)
```

In [90]:

```
ca
```

Out[90]:

```
array([[19098,     0],
       [   29,     6]], dtype=int64)
```

In [91]:

```
print('True positives (TP) = ', cm[0,0])
print('False positives (FP)= ', cm[0, 1])
print('False Negatives (FN)= ', cm[1, 0])
print('True Negatives (TN))= ', cm[1, 1])
```

```
True positives (TP) =  19098
False positives (FP)=  0
False Negatives (FN)=  29
True Negatives (TN))=  6
```

In [92]:

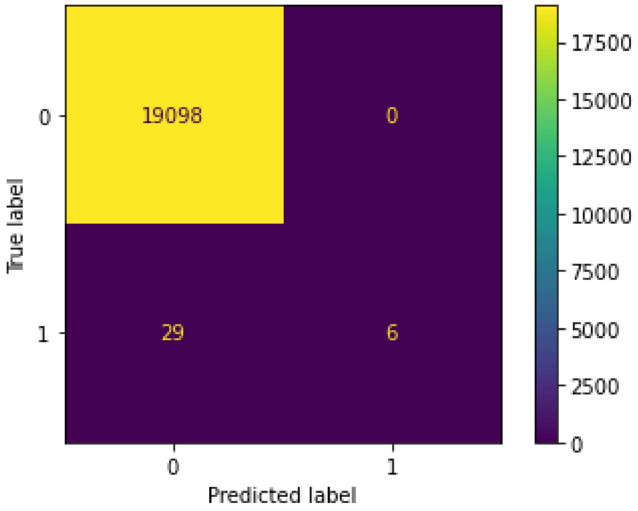
```
#visualze with heatmap
from sklearn.metrics import ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=logreg.classes_)
```

In [93]:

```
disp.plot()
```

Out[93]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x17aff87c
490>
```



In [94]:

```
from sklearn.metrics import classification_report

#classification_report(y_test, y_pred_test)
```

In [95]:

```
#new_df
```

In [42]:

```
#new_df2 = new_df.drop(['CurrencyCode','CountryCode'],axis = 1, inplace=True)
```

In [44]:

```
#print(new_df2)
```

```
None
```

In [ ]:

```
#split data into target and feature

#X = new_df.drop(['CurrencyCode','CountryCode','FraudResult'], axis =1)
#y= new_df['FraudResult']
```

In [96]:

```
print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     19098
           1       1.00      0.17      0.29        35

    accuracy                           1.00     19133
   macro avg       1.00      0.59      0.65     19133
weighted avg       1.00      1.00      1.00     19133
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
#split into test and train data

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [ ]:

```
#check the shapeof our split

#X_train.shape, X_test.shape

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

#X_train = pd.concat([X_train,ProductCategory_dummy, productId_dummy,providerId_dummy,C

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: