In [1]:

```
##Applying Logistic regression
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
import sklearn
import numpy as np

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [2]:

```
df = pd.read_csv("D:\Accredian task\Fraud.csv")
```

In [3]:

```
df.head()
```

Out[3]:

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

In [4]:

```
df.shape
```

Out[4]:

```
(6362620, 11)
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype  
---  ------          -----  
 0   step            int64  
 1   type            object 
 2   amount          float64
 3   nameOrig        object 
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object 
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64  
 10  isFlaggedFraud  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

In [6]:

```
print ('Not Fraud % ',round(df['isFraud'].value_counts()[0]/len(df)*100,2))
print ()
print (round(df.amount[df.isFraud == 0].describe(),2))
print ()
print ()
print ('Fraud %    ',round(df['isFraud'].value_counts()[1]/len(df)*100,2))
print ()
print (round(df.amount[df.isFraud == 1].describe(),2))
```

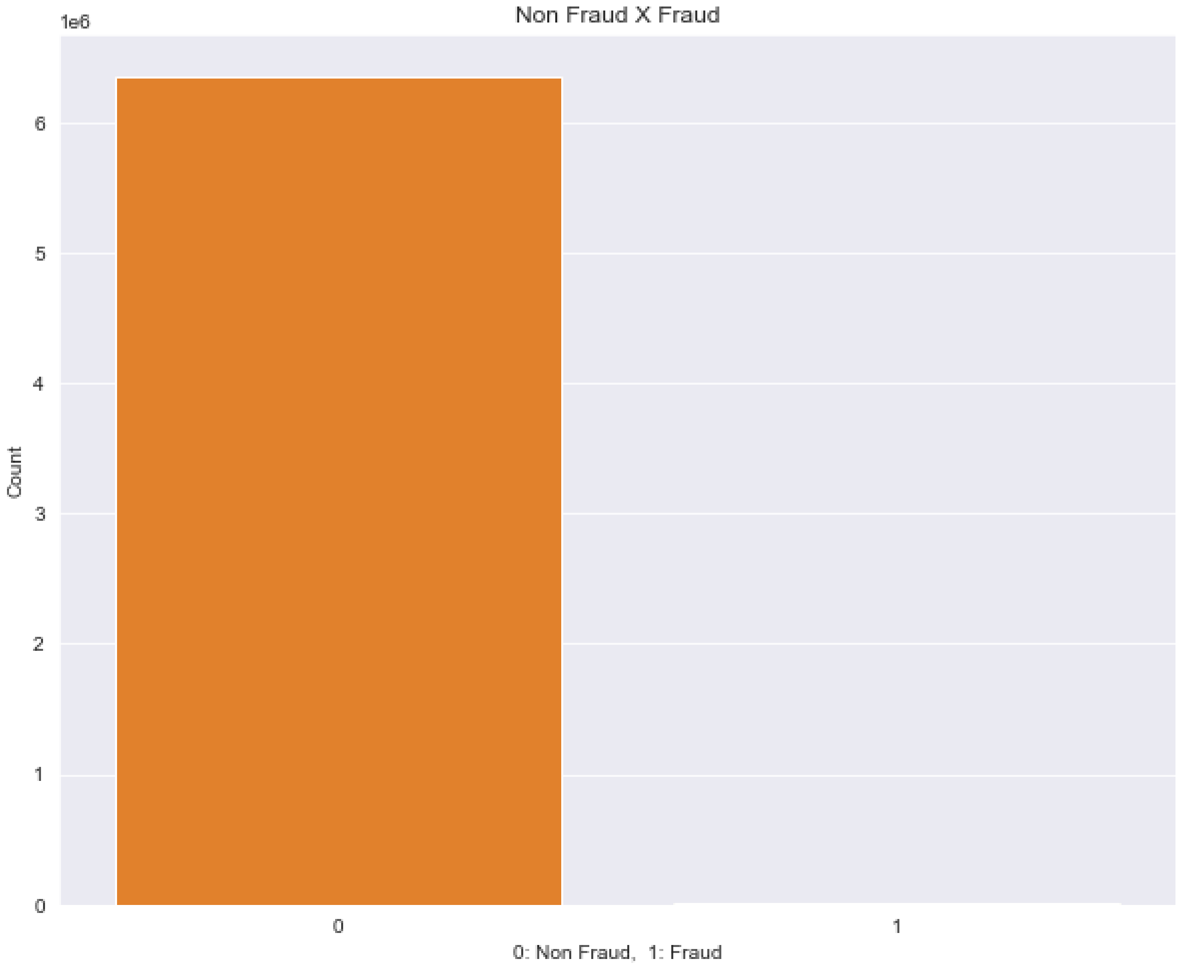
```
Not Fraud %  99.87

count     6354407.00
mean       178197.04
std        596236.98
min             0.01
25%         13368.40
50%         74684.72
75%        208364.76
max      92445516.64
Name: amount, dtype: float64


Fraud %     0.13

count        8213.00
mean      1467967.30
std       2404252.95
min             0.00
25%        127091.33
50%        441423.44
75%       1517771.48
max      10000000.00
Name: amount, dtype: float64
```

Comparing the amount value for normal transaction vs normal

In [7]:

```
plt.figure(figsize=(10,8))
sns.set_style('darkgrid')
sns.barplot(x=df['isFraud'].value_counts().index,y=df['isFraud'].value_counts(), palette
plt.title('Non Fraud X Fraud')
plt.ylabel('Count')
plt.xlabel('0: Non Fraud,  1: Fraud')
print ('Non Fraud % ',round(df['isFraud'].value_counts()[0]/len(df)*100,2))
print ('Fraud %    ',round(df['isFraud'].value_counts()[1]/len(df)*100,2));
```

```
Non Fraud %  99.87
Fraud %     0.13
```



In [8]:

```
##Seperation of input variables from target variables

feature_names = df.iloc[:, 1:9].columns
target = df.iloc[:1, 9:10].columns

data_features = df[feature_names]
data_target = df[target]
```

In [9]:

```
feature_names
```

Out[9]:

```
Index(['type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOrig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest'],
      dtype='object')
```

In [10]:

```
target
```

Out[10]:

```
Index(['isFraud'], dtype='object')
```

In [11]:

```
data_features.head()
```

Out[11]:

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

In [12]:
```
df1 = data_features.drop('type',axis=1)
df1.head(5)
```

Out[12]:

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

In [13]:
```
#data_features.drop('nameOrig',axis=1)
```

In [14]:
```
df2 = df1.drop('nameOrig',axis=1)
df2.head(5)
```

Out[14]:

	amount	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest
0	9839.64	170136.0	160296.36	M1979787155	0.0	0.0
1	1864.28	21249.0	19384.72	M2044282225	0.0	0.0
2	181.00	181.0	0.00	C553264065	0.0	0.0
3	181.00	181.0	0.00	C38997010	21182.0	0.0
4	11668.14	41554.0	29885.86	M1230701703	0.0	0.0

In [15]:
```
df3 = df2.drop('nameDest',axis= 1)
df3.head(5)
```

Out[15]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
0	9839.64	170136.0	160296.36	0.0	0.0
1	1864.28	21249.0	19384.72	0.0	0.0
2	181.00	181.0	0.00	0.0	0.0
3	181.00	181.0	0.00	21182.0	0.0
4	11668.14	41554.0	29885.86	0.0	0.0

In [16]:
```
from sklearn.model_selection import train_test_split
np.random.seed(123)
X_train, X_test, y_train, y_test = train_test_split(df3, data_target,
                                                    train_size = 0.70, test_size = 0.30,
```

In [17]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

In [18]:
```
df.dtypes
```

Out[18]:

```
step                int64
type               object
amount            float64
nameOrig           object
oldbalanceOrg     float64
newbalanceOrig    float64
nameDest           object
oldbalanceDest    float64
newbalanceDest    float64
isFraud             int64
isFlaggedFraud      int64
dtype: object
```

In [19]:

```
lr.fit(X_train, y_train)
```

Out[19]:

```
LogisticRegression()
```

In [20]:

```
def PrintStats(cmat, y_test, pred):
    tpos = cmat[0][0]
    fneg = cmat[1][1]
    fpos = cmat[0][1]
    tneg = cmat[1][0]
```

In [21]:

```
def RunModel(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train.values.ravel())
    pred = model.predict(X_test)
    matrix = confusion_matrix(y_test, pred)
    return matrix, pred
```

In [31]:

```
pip install scikit-plot
```

```
Collecting scikit-plot
  Downloading scikit_plot-0.3.7-py3-none-any.whl (33 kB)
Requirement already satisfied: matplotlib>=1.4.0 in c:\users\nadee\anaconda\lib\site-pac
kages (from scikit-plot) (3.5.1)
Requirement already satisfied: scipy>=0.9 in c:\users\nadee\anaconda\lib\site-packages
(from scikit-plot) (1.7.3)
Requirement already satisfied: joblib>=0.10 in c:\users\nadee\anaconda\lib\site-packages
(from scikit-plot) (1.1.0)
Requirement already satisfied: scikit-learn>=0.18 in c:\users\nadee\anaconda\lib\site-pa
ckages (from scikit-plot) (1.0.2)
Requirement already satisfied: packaging>=20.0 in c:\users\nadee\anaconda\lib\site-packa
ges (from matplotlib>=1.4.0->scikit-plot) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\nadee\anaconda\lib\site-
packages (from matplotlib>=1.4.0->scikit-plot) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\nadee\anaconda\lib\site-pac
kages (from matplotlib>=1.4.0->scikit-plot) (1.3.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\nadee\anaconda\lib\site-pac
kages (from matplotlib>=1.4.0->scikit-plot) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\nadee\anaconda\lib\site-pack
ages (from matplotlib>=1.4.0->scikit-plot) (3.0.4)
Requirement already satisfied: cycler>=0.10 in c:\users\nadee\anaconda\lib\site-packages
(from matplotlib>=1.4.0->scikit-plot) (0.11.0)
Requirement already satisfied: numpy>=1.17 in c:\users\nadee\anaconda\lib\site-packages
(from matplotlib>=1.4.0->scikit-plot) (1.21.5)
Requirement already satisfied: pillow>=6.2.0 in c:\users\nadee\anaconda\lib\site-package
s (from matplotlib>=1.4.0->scikit-plot) (9.0.1)
Requirement already satisfied: six>=1.5 in c:\users\nadee\anaconda\lib\site-packages (fr
om python-dateutil>=2.7->matplotlib>=1.4.0->scikit-plot) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\nadee\anaconda\lib\site-
packages (from scikit-learn>=0.18->scikit-plot) (2.2.0)
Installing collected packages: scikit-plot
Successfully installed scikit-plot-0.3.7
Note: you may need to restart the kernel to use updated packages.
```

In [33]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc
import scikitplot
```

In [34]:

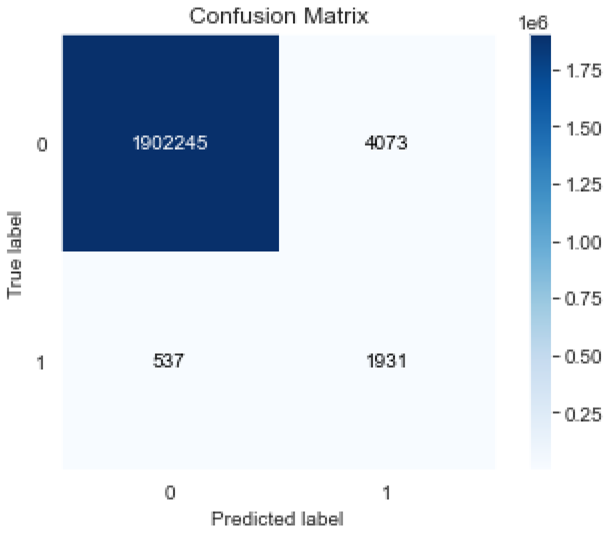
```
cmat, pred = RunModel(lr, X_train, y_train, X_test, y_test)
```

In [35]:

```
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(y_test, pred)
```

Out[35]:

```
<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted label', ylabel='True
label'>
```



In [36]:

```
accuracy_score(y_test, pred)
```

Loading [MathJax]/extensions/Safe.js

Out[36]:

```
0.9975848523616582
```

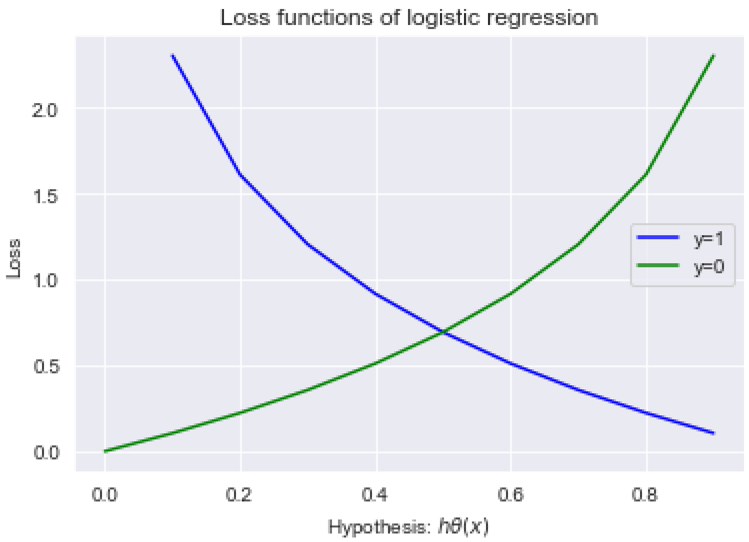
In [37]:

```
print (classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   1906318
           1       0.32      0.78      0.46      2468

    accuracy                           1.00   1908786
   macro avg       0.66      0.89      0.73   1908786
weighted avg       1.00      1.00      1.00   1908786
```

In [60]:

```
xvals = np.arange(0,1,0.1)
y1vals = 0-np.log(xvals)
y0vals = 0-np.log(1-xvals)
plt.plot(xvals, y1vals, 'b', label='y=1')
plt.plot(xvals, y0vals, 'g', label='y=0')
plt.title('Loss functions of logistic regression')
plt.legend()
plt.xlabel('Hypothesis: $h\\theta(x)$')
plt.ylabel('Loss');
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Loading [MathJax]/extensions/Safe.js