In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv("C:/Users/Admin/Downloads/archive/creditcard.csv")
df.sample(5)
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	
225062	144086.0	2.014303	0.222328	-1.730133	0.517234	0.304026	-1.365999	0.364623	-0.437
76060	56379.0	-0.499146	1.225395	2.125937	2.726520	0.248282	0.593994	0.465769	0.206
89720	62698.0	-0.534420	0.918204	2.042266	0.455937	0.032230	0.639126	0.083430	-0.890
134611	80885.0	1.211280	0.639500	-0.452050	0.897389	0.166469	-1.167578	0.360975	-0.218
191073	129094.0	-0.651906	-5.522984	-2.372308	0.216573	-2.216428	0.065825	1.114007	-0.374

5 rows × 31 columns

In [4]:
```
df.Class.value_counts()
```

Out[4]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [5]:
```
X = df.drop('Class',axis='columns')
y = df['Class']
```

In [6]:
```
from imblearn.over_sampling import SMOTE

smote = SMOTE(sampling_strategy='minority')
X_sm, y_sm = smote.fit_resample(X, y)

y_sm.value_counts()
```

Out[6]:
```
0    284315
1    284315
Name: Class, dtype: int64
```

In [7]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_sm, y_sm, test_size=0.2, ra
```

In [8]:
```
# Number of classes in training Data
y_train.value_counts()
```

Out[8]:
```
0    227452
1    227452
Name: Class, dtype: int64
```

In [21]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

In [41]:
```
from sklearn.metrics import accuracy_score
def log_reg(X_train, y_train, X_test, y_test, weights):
    if weights==-1:
        model = LogisticRegression()
    else:
        model = LogisticRegression(class_weight={0:weights[0], 1:weights[1]})

    model.fit(X_train, y_train)
    acc = model.score(X_test, y_test)
    print("Accuracy", acc, "\n")

    y_pred = model.predict(X_test)
    print("preds", y_pred[:5], "\n")

    cl_rep = classification_report(y_test,y_pred)
    print(cl_rep)

    return y_pred
```

In [42]:
```
weights = -1 # pass -1 to use Logistics Regression without weights
log_reg(X_train, y_train, X_test, y_test, weights)
```

```
Accuracy 0.9739725304679668

preds [0 0 1 0 1]

              precision    recall  f1-score   support

           0       0.97      0.98      0.97     56863
           1       0.98      0.96      0.97     56863

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726
```

Out[42]:
```
array([0, 0, 1, ..., 1, 1, 0], dtype=int64)
```

In [43]:
```
from sklearn.metrics import confusion_matrix
y_predict = log_reg(X_train, y_train, X_test, y_test, weights)
cnf_matrix = confusion_matrix(y_test, y_predict)
print('Confusion matrix:')
print(cnf_matrix)
```

```
Accuracy 0.9739725304679668

preds [0 0 1 0 1]

              precision    recall  f1-score   support

           0       0.97      0.98      0.97     56863
           1       0.98      0.96      0.97     56863

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726

Confusion matrix:
[[55904   959]
 [ 2001 54862]]
```

In [ ]: