In [86]:
```
import pandas as pd
import numpy as np
import tensorflow
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import xlrd
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [87]:
```
#dataframe1 = pd.read_excel('test_data.xlsx')
test_data = pd.read_csv("test_data.csv", encoding = "latin1")
train_data = pd.read_csv("train_data.csv", encoding = "latin1")
test_data_hidden = pd.read_csv("test_data_hidden.csv", encoding = "latin1")
```

In [88]:
```
test_data.shape
```

Out[88]:
```
(56962, 30)
```

In [89]:
```
train_data.shape
```

Out[89]:
```
(227845, 31)
```

In [90]:
```
test_data_hidden.shape
```

Out[90]:
```
(56962, 31)
```

In [91]:
```
test_data.head(5)
```

Out[91]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	113050.0	0.114697	0.796303	-0.149553	-0.823011	0.878763	-0.553152	0.939259	-0.108502
1	26667.0	-0.039318	0.495784	-0.810884	0.546693	1.986257	4.386342	-1.344891	-1.743736
2	159519.0	2.275706	-1.531508	-1.021969	-1.602152	-1.220329	-0.462376	-1.196485	-0.147058
3	137545.0	1.940137	-0.357671	-1.210551	0.382523	0.050823	-0.171322	-0.109124	-0.002115
4	63369.0	1.081395	-0.502615	1.075887	-0.543359	-1.472946	-1.065484	-0.443231	-0.143374

5 rows × 30 columns

In [92]:
```
train_data.head(5)
```

Out[92]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	38355.0	1.043949	0.318555	1.045810	2.805989	-0.561113	-0.367956	0.032736	-0.042333
1	22555.0	-1.665159	0.808440	1.805627	1.903416	-0.821627	0.934790	-0.824802	0.975890
2	2431.0	-0.324096	0.601836	0.865329	-2.138000	0.294663	-1.251553	1.072114	-0.334896
3	86773.0	-0.258270	1.217501	-0.585348	-0.875347	1.222481	-0.311027	1.073860	-0.161408
4	127202.0	2.142162	-0.494988	-1.936511	-0.818288	-0.025213	-1.027245	-0.151627	-0.305750

5 rows × 31 columns

In [93]:
```
test_data_hidden.head(5)
```

Out[93]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	113050.0	0.114697	0.796303	-0.149553	-0.823011	0.878763	-0.553152	0.939259	-0.108502
1	26667.0	-0.039318	0.495784	-0.810884	0.546693	1.986257	4.386342	-1.344891	-1.743736
2	159519.0	2.275706	-1.531508	-1.021969	-1.602152	-1.220329	-0.462376	-1.196485	-0.147058
3	137545.0	1.940137	-0.357671	-1.210551	0.382523	0.050823	-0.171322	-0.109124	-0.002115
4	63369.0	1.081395	-0.502615	1.075887	-0.543359	-1.472946	-1.065484	-0.443231	-0.143374

5 rows × 31 columns

In [94]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [95]:
```
test_data.isnull().sum()
```

Out[95]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
dtype: int64
```

In [96]:
```
train_data.isnull().sum()
```

Out[96]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

In [97]:
```
test_data_hidden.isnull().sum()
```

Out[97]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

In [98]:
```
# distribution of legit transaction and fraudulent transaction
train_data['Class'].value_counts()
```

Out[98]:
```
0    227451
1       394
Name: Class, dtype: int64
```

In [99]:
```
legit = train_data[train_data.Class == 0]
fraud = train_data[train_data.Class == 1]

print('legit shape', legit.shape)
print('fraud shape', fraud.shape)
```

```
legit shape (227451, 31)
fraud shape (394, 31)
```

In [100…
```
legit_sample= legit.sample(n=394)

new_test_data = pd.concat([legit_sample ,fraud], axis=0)

print(new_test_data.shape)
```

```
(788, 31)
```

In [101…
```
new_test_data['Class'].value_counts()
```

Out[101]:
```
0    394
1    394
Name: Class, dtype: int64
```

In [102…
```
X = new_test_data.drop(columns='Class',axis=1)
Y = new_test_data['Class']
```

```
X_train, X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.2 , stratify
```

In [103…]
```
print(X.shape,X_train.shape , X_test.shape)
```

```
(788, 30) (630, 30) (158, 30)
```

In [104…]
```
# training model
model = LogisticRegression()
```

In [105…]
```
#training the logistic regression model with training data

model.fit(X_train,Y_train)
```

Out[105]:
```
▾ LogisticRegression

LogisticRegression()
```

In [106…]
```
#accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction , Y_train)
```

In [107…]
```
print('accuracy on training data :' , training_data_accuracy )
```

```
accuracy on training data : 0.9428571428571428
```

In [108…]
```
#accuracy on testing data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction , Y_test)
```

In [109…]
```
print('accuracy on testing data :' , test_data_accuracy )
```

```
accuracy on testing data : 0.9367088607594937
```

In [ ]: