In [1]:
```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

In [2]:
```
df=pd.read_excel("f:/dataset/classification/creditcard_copy.xlsx")
X=df.iloc[:,1:-2].values
y=df.iloc[:,-1].values
```

In [3]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1)
model=KNeighborsClassifier()
model.fit(X_train,y_train)
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.9855567226890757
0.9921259842519685
```

In [4]:
```
#Confusion Matrix---->it shows model behaviour for each class
from sklearn.metrics import confusion_matrix
pred_train=model.predict(X_train)
confusion_matrix(y_train,pred_train) #recommended
```

Out[4]:
```
array([[3420,    4],
       [  51,  333]], dtype=int64)
```

In [5]:
```
confusion_matrix(pred_train,y_train)
```

Out[5]:
```
array([[3420,   51],
       [   4,  333]], dtype=int64)
```

In [6]:
```
confusion_matrix(y_train,pred_train)
```

Out[6]:
```
array([[3420,    4],
       [  51,  333]], dtype=int64)
```

In [13]:
```
p_1=333/(333+4)
print(p_1)
p_0=3420/(3420+51)
print(p_0)
from sklearn.metrics import precision_score,recall_score,accuracy_score
print(precision_score(y_train,pred_train))
print(precision_score(y_train,pred_train,average=None))
r_0=3420/(3420+4)
print(r_0)

r_1=333/(51+333)
print(r_1)
print(recall_score(y_train,pred_train,average=None))
acc_sc=(3420+333)/(3420+4+51+333)
print(acc_sc)
print(accuracy_score(y_train,pred_train))
```

```
0.9881305637982196
0.9853068280034573
0.9881305637982196
[0.98530683 0.98813056]
0.9988317757009346
0.8671875
[0.99883178 0.8671875 ]
0.9855567226890757
0.9855567226890757
```

In [14]:
```
from sklearn.metrics import classification_report
print(classification_report(y_train,pred_train))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      3424
           1       0.99      0.87      0.92       384

    accuracy                           0.99      3808
   macro avg       0.99      0.93      0.96      3808
weighted avg       0.99      0.99      0.99      3808
```

In [15]:
```
print(classification_report(y_test,model.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1162
           1       0.98      0.93      0.95       108

    accuracy                           0.99      1270
   macro avg       0.99      0.96      0.97      1270
weighted avg       0.99      0.99      0.99      1270
```

In [ ]:
```

```