```
Task-5 CREDIT CARD FRAUD DETECTION

Author: Bandana Prakash
Batch :June
Domain: Data Science
Aim: Build a machine learning model to identify fraudulent credit card transactions.
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:
```
file=pd.read_csv("creditcard.csv")
```

In [3]:
```
file.head(10)
```

Out[3]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V21 | V22 | V23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | ... | -0.018307 | 0.277838 | -0.110474 |
| **1** | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | ... | -0.225775 | -0.638672 | 0.101288 |
| **2** | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | ... | 0.247998 | 0.771679 | 0.909412 |
| **3** | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | ... | -0.108300 | 0.005274 | -0.190321 |
| **4** | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | ... | -0.009431 | 0.798278 | -0.137458 |
| **5** | 2.0 | -0.425966 | 0.960523 | 1.141109 | -0.168252 | 0.420987 | -0.029728 | 0.476201 | 0.260314 | -0.568671 | ... | -0.208254 | -0.559825 | -0.026398 |
| **6** | 4.0 | 1.229658 | 0.141004 | 0.045371 | 1.202613 | 0.191881 | 0.272708 | -0.005159 | 0.081213 | 0.464960 | ... | -0.167716 | -0.270710 | -0.154104 |
| **7** | 7.0 | -0.644269 | 1.417964 | 1.074380 | -0.492199 | 0.948934 | 0.428118 | 1.120631 | -3.807864 | 0.615375 | ... | 1.943465 | -1.015455 | 0.057504 |
| **8** | 7.0 | -0.894286 | 0.286157 | -0.113192 | -0.271526 | 2.669599 | 3.721818 | 0.370145 | 0.851084 | -0.392048 | ... | -0.073425 | -0.268092 | -0.204233 |
| **9** | 9.0 | -0.338262 | 1.119593 | 1.044367 | -0.222187 | 0.499361 | -0.246761 | 0.651583 | 0.069539 | -0.736727 | ... | -0.246914 | -0.633753 | -0.120794 |

10 rows × 31 columns

In [4]:
```
file.describe()
```

Out[4]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 284807.000000 | 2.848070e+05 | 2.848070e+05 | 2.848070e+05 | 2.848070e+05 | 2.848070e+05 | 2.848070e+05 | 2.848070e+05 | 2.848070e+0 |
| **mean** | 94813.859575 | 1.168375e-15 | 3.416908e-16 | -1.379537e-15 | 2.074095e-15 | 9.604066e-16 | 1.487313e-15 | -5.556467e-16 | 1.213481e-1 |
| **std** | 47488.145955 | 1.958696e+00 | 1.651309e+00 | 1.516255e+00 | 1.415869e+00 | 1.380247e+00 | 1.332271e+00 | 1.237094e+00 | 1.194353e+0 |
| **min** | 0.000000 | -5.640751e+01 | -7.271573e+01 | -4.832559e+01 | -5.683171e+00 | -1.137433e+02 | -2.616051e+01 | -4.355724e+01 | -7.321672e+0 |
| **25%** | 54201.500000 | -9.203734e-01 | -5.985499e-01 | -8.903648e-01 | -8.486401e-01 | -6.915971e-01 | -7.682956e-01 | -5.540759e-01 | -2.086297e-0 |
| **50%** | 84692.000000 | 1.810880e-02 | 6.548556e-02 | 1.798463e-01 | -1.984653e-02 | -5.433583e-02 | -2.741871e-01 | 4.010308e-02 | 2.235804e-0 |
| **75%** | 139320.500000 | 1.315642e+00 | 8.037239e-01 | 1.027196e+00 | 7.433413e-01 | 6.119264e-01 | 3.985649e-01 | 5.704361e-01 | 3.273459e-0 |
| **max** | 172792.000000 | 2.454930e+00 | 2.205773e+01 | 9.382558e+00 | 1.687534e+01 | 3.480167e+01 | 7.330163e+01 | 1.205895e+02 | 2.000721e+0 |

8 rows × 31 columns

In [5]:
```
file.isnull().sum()
```

Out[5]:
```
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 [6]:
```
file['Class'].value_counts()
```

Out[6]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [7]:
```
normal=file[file.Class==0]
```

In [8]:
```
fraud=file[file.Class==1]
```

In [9]:
```
print(normal.shape)
```

```
(284315, 31)
```

In [10]:
```
print(fraud.shape)
```

```
(492, 31)
```

In [11]:
```
normal.Amount.describe()
```

Out[11]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

In [12]:
```
fraud.Amount.describe()
```

Out[12]:
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

In [13]:
```
file.groupby('Class').mean()
```

Out[13]:

| Class | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V20 | V2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 94838.202258 | 0.008258 | -0.006271 | 0.012171 | -0.007860 | 0.005453 | 0.002419 | 0.009637 | -0.000987 | 0.004467 | ... | -0.000644 | -0.00123 |
| **1** | 80746.806911 | -4.771948 | 3.623778 | -7.033281 | 4.542029 | -3.151225 | -1.397737 | -5.568731 | 0.570636 | -2.581123 | ... | 0.372319 | 0.71358 |

2 rows × 30 columns

In [14]:
```
normal_sample=normal.sample(n=492)
```

In [15]:
```
new_file=pd.concat([normal_sample,fraud],axis=0)
```

In [16]:
```
new_file.head(10)
```

Out[16]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V21 | V22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **220813** | 142310.0 | 0.530996 | -2.825616 | -1.736751 | 0.449577 | -0.135923 | 2.047654 | 0.162387 | 0.428858 | 0.848089 | ... | 0.071907 | -1.212468 |
| **67440** | 52535.0 | -0.930321 | 0.758878 | 1.455930 | 1.727768 | 0.410741 | 0.371296 | 0.851601 | 0.347836 | -0.916291 | ... | 0.125086 | 0.228744 |
| **214324** | 139625.0 | 2.040578 | -0.146368 | -2.955721 | -0.578510 | 2.609546 | 3.142573 | -0.417135 | 0.784442 | 0.359925 | ... | -0.352449 | -0.996836 |
| **4479** | 3781.0 | 1.031446 | -0.026018 | 2.416341 | 3.272713 | -1.232136 | 1.043872 | -1.247290 | 0.307633 | 2.278284 | ... | -0.070100 | 0.562415 |
| **42789** | 41245.0 | 0.897318 | -0.611802 | 1.411445 | 1.554270 | -1.047296 | 0.921290 | -0.778799 | 0.359397 | 1.263493 | ... | -0.085606 | 0.030504 |
| **190353** | 128797.0 | -0.977643 | -0.478000 | -0.467955 | -0.211594 | 2.469079 | -1.719093 | 0.387356 | -0.120749 | -0.327616 | ... | -0.040164 | -0.545121 |
| **236849** | 148987.0 | 1.242204 | -0.498560 | -1.176075 | 3.954839 | 0.361610 | 0.254794 | 0.772584 | -0.183045 | -1.094441 | ... | -0.062080 | -1.013477 |
| **271916** | 164808.0 | 2.020674 | 0.141273 | -1.595421 | 0.341551 | 0.414815 | -0.655331 | 0.109926 | -0.121655 | 0.167229 | ... | -0.296948 | -0.767893 |
| **198636** | 132579.0 | -0.641993 | -1.294167 | 2.094085 | -2.961239 | -1.389647 | 0.538700 | -0.855272 | 0.357768 | -1.669708 | ... | 0.265813 | 0.879345 |
| **177250** | 123100.0 | -2.779551 | 0.203259 | -3.155954 | -2.913320 | 2.721824 | 2.327317 | -0.398472 | 1.719109 | -1.713604 | ... | -0.202913 | -0.105620 |

10 rows × 31 columns

In [17]:
```
new_file['Class'].value_counts()
```

Out[17]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

In [18]:
```
new_file.groupby('Class').mean()
```

Out[18]:

| Class | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V20 | V21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 91325.664634 | -0.044825 | 0.029433 | 0.082440 | -0.005140 | -0.032174 | 0.051381 | 0.009840 | -0.033033 | 0.023252 | ... | -0.032774 | -0.069963 |
| **1** | 80746.806911 | -4.771948 | 3.623778 | -7.033281 | 4.542029 | -3.151225 | -1.397737 | -5.568731 | 0.570636 | -2.581123 | ... | 0.372319 | 0.713588 |

2 rows × 30 columns

In [19]:
```
X=new_file.drop(columns='Class',axis=1)
```

In [20]:
```
Y=new_file['Class']
```

In [21]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

In [22]:
```
model=LogisticRegression()
```

In [23]:
```
model.fit(X_train,Y_train)
```

```
/Volumes/Prototype/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:460: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/module
s/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-lear
n.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[23]:
```
LogisticRegression()
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [24]:
```
X_train_prediction=model.predict(X_train)
```

In [25]:
```
training_data_acuracy=accuracy_score(X_train_prediction,Y_train)*100
```

In [26]:
```
print(f"Training Data Accuracy: {training_data_acuracy}%")
```

```
Training Data Accuracy: 93.90088945362135%
```

In [27]:
```
X_test_prediction=model.predict(X_test)
```

In [28]:
```
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)*100
```

In [29]:
```
print(f"Test Data Accuracy: {test_data_accuracy}%")
```

```
Test Data Accuracy: 91.87817258883248%
```