# CREDIT CARD FRAUD DETECTION

# DEVELOPMENT PART-1

## ► Importing the required libraries

Let's start the development part of credit card fraud detection with machine learning by importing the necessary Python libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
```

## ► Importing the data set

## Loading and Understanding the data

```
data=pd.read_csv("/content/creditcard.csv")
print(data.head())
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | |
| 1 | 0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | |
| 2 | 1 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | |
| 3 | 1 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | |
| 4 | 2 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | |

| | V8 | V9 | ... | V21 | V22 | V23 | V24 | V25 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.098698 | 0.363787 | ... | -0.018307 | 0.277838 | -0.110474 | 0.066928 | 0.128539 | |
| 1 | 0.085102 | -0.255425 | ... | -0.225775 | -0.638672 | 0.101288 | -0.339846 | 0.167170 | |
| 2 | 0.247676 | -1.514654 | ... | 0.247998 | 0.771679 | 0.909412 | -0.689281 | -0.327642 | |
| 3 | 0.377436 | -1.387024 | ... | -0.108300 | 0.005274 | -0.190321 | -1.175575 | 0.647376 | |
| 4 | -0.270533 | 0.817739 | ... | -0.009431 | 0.798278 | -0.137458 | 0.141267 | -0.206010 | |

| | V26 | V27 | V28 | Amount | Class |
|---|---|---|---|---|---|
| 0 | -0.189115 | 0.133558 | -0.021053 | 149.62 | 0.0 |
| 1 | 0.125895 | -0.008983 | 0.014724 | 2.69 | 0.0 |
| 2 | -0.139097 | -0.055353 | -0.059752 | 378.66 | 0.0 |
| 3 | -0.221929 | 0.062723 | 0.061458 | 123.50 | 0.0 |
| 4 | 0.502292 | 0.219422 | 0.215153 | 69.99 | 0.0 |

[5 rows x 31 columns]

## ►Handling the missing data

◉Handling missing data refers to the process of managing and addressing the absence of information or values in a dataset.

◉It involves various techniques and strategies to deal with data points that are either incomplete or entirely missing

◉The objective is to make the dataset suitable for analysis, modeling, or other data-driven tasks while minimizing the potential biases and errors introduced by the missing data.

◉Common approaches for handling missing data include imputation, deletion, or the use of statistical methods to estimate or replace the missing values.

◉The choice of method depends on the specific dataset, the nature of the missingness, and the goals of the analysis. Proper handling of missing data is essential to ensure the accuracy and reliability of data-driven insights and decision-making.

```
print(data.isnull().sum())
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
```

## ►Describing the data

```
print(data.shape)
print(data.describe())
```

| | Time | V1 | ... | Amount | Class |
|---|---|---|---|---|---|
| count | 284807.000000 | 2.848070e+05 | ... | 284807.000000 | 284807.000000 |
| mean | 94813.859575 | 3.919560e-15 | ... | 88.349619 | 0.001727 |
| std | 47488.145955 | 1.958696e+00 | ... | 250.120109 | 0.041527 |
| min | 0.000000 | -5.640751e+01 | ... | 0.000000 | 0.000000 |
| 25% | 54201.500000 | -9.203734e-01 | ... | 5.600000 | 0.000000 |
| 50% | 84692.000000 | 1.810880e-02 | ... | 22.000000 | 0.000000 |
| 75% | 139320.500000 | 1.315642e+00 | ... | 77.165000 | 0.000000 |
| max | 172792.000000 | 2.454930e+00 | ... | 25691.160000 | 1.000000 |

## ► Encoding Categorical Data

◉Encoding categorical data using one- hot encoding (OneHotEncoder) is a process of representing categorical variables in a numerical format that can be used in various machine learning algorithms

◉Encoding categorical data using OneHotEncoder is a process in data projects where categorical variables are transformed into a numerical format, specifically binary vectors.

◉Each unique category or label within a categorical variable is converted into a separate binary column, and for each observation, the column corresponding to its category is marked with a “1,” while all other columns are set to “O.”

◉This method is used to make categorical data compatible with machine learning algorithms that require numerical input.

◉OneHotEncoder ensures that the categorical data is represented in a way that doesn’t introduce false ordinal relationships or numerical values, preventing biases in the model’s interpretation of the data

## ►Splitting the data set

**Testing set :**The testing set is a smaller portion of the data, usually around 20-30% of the dataset. It is kept separate and is not used during the model training phase. Instead, it is used to evaluate the

model's performance by making predictions or performing analyses and comparing them to the actual, known outcomes.

**Training set :**This subset contains a majority of the data, typically around 70-80% of the dataset. It is used to train machine learning models or perform data analysis tasks. The model learns patterns, relationships, and trends within the data from this set.

```
X = data.drop(['Class'], axis = 1)
Y = data["Class"]
print(X.shape)
print(Y.shape)

xData = X.values
yData = Y.values
```

```
from sklearn.model_selection import train_test_split

xTrain, xTest, yTrain, yTest = train_test_split(
        xData, yData, test_size = 0.2, random_state = 42)
```

## ▸Feature Scaling

◉Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.

◉If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

## Standardization

▸This method of scaling is basically based on the central tendencies and variance of the data.

◉First, we should calculate the mean and standard deviation of the data we would like to normalize.

◉Then we are supposed to subtract the mean value from each entry and then divide the result by the standard deviation.

import StandardScalar

```
from sklearn.preprocessing import StandardScaler
sc_X=StandardScaler()
Xtrain=sc_X.fit_transform(Xtrain)
Xtest=sc_X.transform(Xtest)
Xtrain
```

So this is how we can train a machine learning model for credit card fraud detection.