**USECASE:5 - Credit Card Fraud Detection**

**AIM**: Train a simple classifier to detect credit-card fraud on an imbalanced dataset (fraud ≈ 1%). Report accuracy, precision, recall, F1 and confusion matrix.

**ALGORITHM:**

1.Generate an imbalanced dataset that simulates fraud (1% positive class).

2. Split into train/test sets.

3. Scale features (StandardScaler).

4. Train a classifier (Logistic Regression with balanced class weights).

5. Predict on test set and compute evaluation metrics (accuracy, precision, recall, F1, confusion matrix).

6. Print results.

**CODE:**

# Run this in Google Colab

!pip -q install scikit-learn --upgrade

import numpy as np

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report

# 1) Create synthetic imbalanced dataset (simulate credit-card fraud)

X, y = make\_classification(n\_samples=10000, n\_features=20, n\_informative=6, n\_redundant=2,

n\_clusters\_per\_class=1, weights=[0.99], flip\_y=0.01, class\_sep=1.0, random\_state=42)

# 2) Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, stratify=y, random\_state=42)

# 3) Scale

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# 4) Train classifier (Logistic Regression with class\_weight to handle imbalance)

clf = LogisticRegression(max\_iter=1000, class\_weight='balanced', solver='liblinear', random\_state=42)

clf.fit(X\_train, y\_train)

# 5) Predict and evaluate

y\_pred = clf.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

prec = precision\_score(y\_test, y\_pred, zero\_division=0)

rec = recall\_score(y\_test, y\_pred, zero\_division=0)

f1 = f1\_score(y\_test, y\_pred, zero\_division=0)

cm = confusion\_matrix(y\_test, y\_pred)

print("=== Evaluation ===")

print(f"Accuracy : {acc:.4f}")

print(f"Precision: {prec:.4f}")

print(f"Recall : {rec:.4f}")

print(f"F1-score : {f1:.4f}")

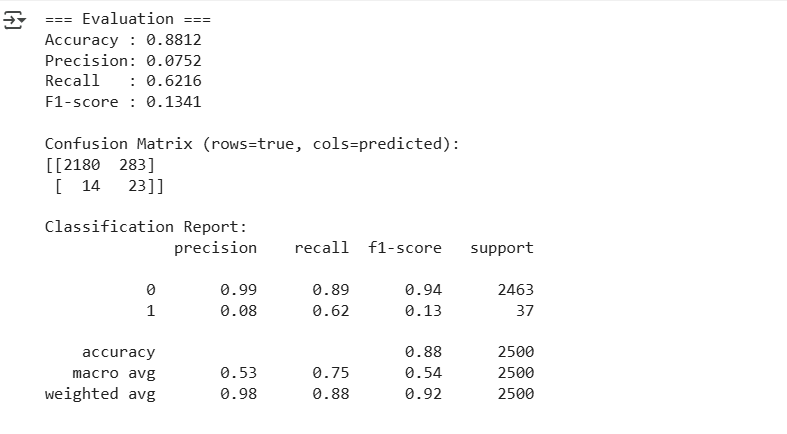
print("\nConfusion Matrix (rows=true, cols=predicted):")

print(cm)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred, zero\_division=0))

**OUTPUT:**

****

**RESULT:** Thus the above program was executed successfully and output was verified.