Experiment – 1.4

**Aim:**

Write a program to implement Fraud Detection in Financial Transactions using Logistic Regression in Business Intelligence

**Software Required:**

* Google Colab
* OR Jupyter Notebook

**Description:**

The experiment "Fraud Detection in Financial Transactions using Logistic Regression in Business Intelligence" aims to develop a program that utilizes logistic regression to detect fraudulent activities in financial transactions. Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class or not. It is a kind of statistical algorithm, which analyse the relationship between a set of independent variables and the dependent binary variables. It is a powerful tool for decision-making. For example email spam or not. By employing logistic regression in business intelligence, we aim to identify patterns and indicators of fraud based on historical transaction data and build a predictive model for future fraud detection.

**Pseudo code/Algorithms/Flowchart/Steps:**

1. Dataset Import and Connection
2. Dataset Preparation
3. Exploratory Data Analysis
4. Programming Environment Setup
5. Implementing Logistic Regression
6. Model Evaluation and Interpretation
7. Fraud Detection
8. Experimentation and Analysis

**Implementation:**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split, learning\_curve

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

ccData = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/creditcard.csv')

ccData.head()

ccData.info()

ccData.isnull().sum()

X = ccData.drop('Class', axis=1)

Y = ccData['Class']

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=50)

scaler = StandardScaler()

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

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

model = LogisticRegression()

model.fit(X\_train, Y\_train)

X\_train\_pred = model.predict(X\_train)

train\_accuracy = accuracy\_score(Y\_train, X\_train\_pred)

print(f'Accuracy score on training data = {train\_accuracy\*100:.2f}%')

X\_test\_pred = model.predict(X\_test)

test\_accuracy = accuracy\_score(Y\_test, X\_test\_pred)

print(f'Accuracy score on test data = {test\_accuracy\*100:.2f}%')

train\_sizes, train\_scores, test\_scores = learning\_curve(model, X\_train, Y\_train, cv=5, scoring='accuracy', train\_sizes=np.linspace(0.1, 1.0, 10))

plt.figure(figsize=(6, 4))

plt.plot(train\_sizes, np.mean(train\_scores, axis=1)\*100, label='Training Score')

plt.plot(train\_sizes, np.mean(test\_scores, axis=1)\*100, label='Testing Score')

plt.xlabel('Dataset Size')

plt.ylabel('Accuracy %')

plt.title('Learning Curves')

plt.legend()

plt.show()

**Output:**

****





