Experiment – 1.4

Aim: Write a program to implement Fraud Detection in Financial Transactions using Logistic Regression.

Software Required: Jupyter Notebook.

Description:

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.

Logistic Function (Sigmoid Function):

The sigmoid function is a mathematical function used to map the predicted values to probabilities.

It maps any real value into another value within a range of 0 and 1. o The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form.

The S-form curve is called the Sigmoid function or the logistic function.

In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Implementation and Output:

- **Import Library**

Course Name: Business Intelligence Lab **Course Code:** CSP-421

```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from scipy.special import expit
```

- **Data Ingestion**

```
data = pd.read_csv("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]
```

- **Removing NaN values**

```
data.dropna(inplace=True)
```

```
[ ] 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
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
```

```
legit = data[data.Class == 0]
fraud = data[data.Class == 1]
```

```
[ ] print(legit.shape)
    print(fraud.shape)

(31677, 31)
(102, 31)
```

```
[ ] legit.Amount.describe()

count    31677.000000
mean        81.082407
std        223.072655
min          0.000000
25%          6.870000
50%         20.000000
75%         73.610000
max       7879.420000
Name: Amount, dtype: float64
```

```
fraud.Amount.describe()
```

```
count      102.000000
mean        91.237451
std        248.270971
min          0.000000
25%          1.000000
50%          3.440000
75%         99.990000
max       1809.680000
Name: Amount, dtype: float64
```

```
legit_sample = legit.sample(n=102)
```

```
new_data = pd.concat([legit_sample, fraud], axis=0)
new_data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
23677	32865	-0.591401	0.822773	1.593352	-0.181900	0.819762	-0.485332	1.044838	-0.193906	-0.675244	...
367	268	1.146065	0.285853	0.562439	1.459336	-0.225891	-0.346303	0.131988	-0.085179	0.136365	...
3422	2930	-0.323191	0.506952	0.353537	-2.597668	1.184509	0.259516	0.899799	0.000528	0.814631	...
30694	36042	-1.592063	0.997902	2.422148	2.625269	0.108126	1.323651	0.022737	0.079881	-0.292401	...
7752	10795	1.083799	0.292250	0.495371	1.387992	-0.199535	-0.613917	0.135991	-0.221068	1.204113	...

5 rows × 31 columns

```
new_data['Class'].value_counts()
```

```
0.0    102
1.0    102
Name: Class, dtype: int64
```

```
X = new_data.drop(columns='Class', axis=1)
Y = new_data['Class']
```

- **Model Trainer**

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

```
[ ] model = LogisticRegression()
    model.fit(X_train, Y_train)

    /usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs
    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
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
      n_iter_i = _check_optimize_result(
    ▾ LogisticRegression
    LogisticRegression()
```

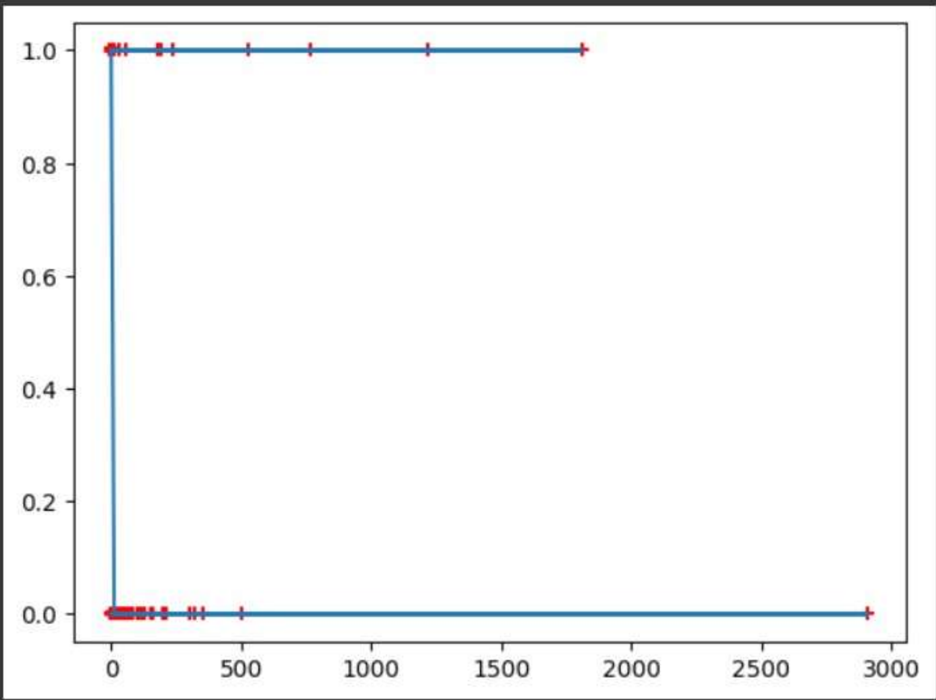
- **Model Accuracy**

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy on Training data : ', training_data_accuracy)

X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score on Test Data : ', test_data_accuracy)

Accuracy on Training data :  0.9631901840490797
Accuracy score on Test Data :  0.9512195121951219
```

- **Regression Plot**

Course Name: Business Intelligence Lab **Course Code:** CSP-421

```
[63] x = new_data['Amount']
     y = new_data['Class']
     plt.scatter(x,y,marker = '+',color = 'red')
     plt.plot(x,y)
     plt.show()
```



Learning Outcome:

1. Understand the concepts and principles of logistic regression as a predictive modelling technique.
2. Gain proficiency in programming and implementing logistic regression algorithms.
3. Apply data pre-processing techniques for preparing the dataset for regression analysis.
4. Perform feature selection and engineering to enhance the predictive power of the model.
5. Assess and interpret the performance of the logistic regression model.