CREDIT CARD FRAUD DETECTION

INTRODUCTION:

Credit card detection using logistic regression is a machine learning technique that can be used to identify fraudulent transactions based on the features of the credit card data. Logistic regression is a type of classification algorithm that predicts the probability of an event (such as fraud) based on a linear combination of input variables (such as transaction amount, time, location, etc.). Some of the benefits of using logistic regression for credit card fraud detection are:

- **Simple and interpretable**: Logistic regression is easy to implement and understand, and it can provide insights into the importance of each feature for fraud detection.
- **Scalable and efficient**: Logistic regression can handle large and high-dimensional datasets with relatively low computational cost and memory usage.
- **Flexible and adaptable**: Logistic regression can be modified and improved by using different techniques, such as feature selection, feature engineering, regularization, sampling, and threshold tuning.

Some of the challenges of using logistic regression for credit card fraud detection are:

- **Imbalanced data**: The credit card data is usually highly imbalanced, meaning that the number of fraudulent transactions is much smaller than the number of normal transactions. This can cause the logistic regression model to be biased towards the majority class and fail to detect the minority class.
- **Non-linear relationships**: The credit card data may contain complex and non-linear relationships between the features and the target variable, which the logistic regression model may not be able to capture accurately.
- **Optimal threshold selection**: The logistic regression model outputs a probability score for each transaction, which needs to be converted into a binary label (fraud or not fraud) by using a threshold value. Choosing the optimal threshold value is a crucial step in the model evaluation process, as it affects the trade-off between false positives and false negatives.

CREDIT CARD FRAUD DETECTION

LOGISTIC REGRESSION:

Logistic regression is a type of **supervised machine learning algorithm** that is used for **classification tasks**. It is used to predict the probability of an event (such as fraud) based on a linear combination of input variables (such as transaction amount, time, location, etc.). Logistic regression is a powerful tool for decision-making because it can provide insights into the importance of each feature for fraud detection. It is mainly used for binary classification where we use a logistic function, also known as a sigmoid function that takes input as independent variables and produces a probability value between 0 and 1.

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. 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 . Logistic regression is much similar to the linear regression except that how they are used. Linear regression is used for solving regression problems, whereas logistic regression is used for solving the classification problems. Logistic regression can handle large and high-dimensional datasets with relatively low computational cost and memory usage. However, some of the challenges of using logistic regression for credit card fraud detection are imbalanced data, non-linear relationships, and optimal threshold selection .

CREDIT CARD FRAUD DETECTION

PROGRAM

```
#import the files
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

#loding the data
credit_card_data=pd.read_csv('/content/drive/MyDrive/creditcard.csv')

#first five rows
credit_card_data.head()

#last five rows
credit_card_data.tail()

#dataset information
credit_card_data.info()

#checking the number of missing data
credit_card_data.isnull().sum()

#distribution of legit transaction and fraudulent transaction
credit_card_data['Class'].value_counts()
```

```
>>  0    284315
    1       492
    Name: Class, dtype: int64
```

CREDIT CARD FRAUD DETECTION

```
"""the above data set unbalanced data
here 0 represent normal transaction
and 1 represent fraudulent transaction"""

#seperating the data
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
print(legit.shape)
print(fraud.shape)
>>  (284315, 31)
    (492, 31)
#statistical measuring of the data
legit.Amount.describe()
fraud.Amount.describe()

#compare the values for both transactions
credit_card_data.groupby('Class').mean()

"""sample data set containimg semilae discribution for normal transaction and fraudulent
transaction number for fraud transaction - 492"""

legit_sample = legit.sample(n=492)
"""concatenating the wo data frame"""

new_dataset = pd.concat([legit_sample,fraud], axis=0)
new_dataset.head()
new_dataset.tail()
new_dataset['Class'].value_counts()
new_dataset.groupby('Class').mean()
```

CREDIT CARD FRAUD DETECTION

```
"""splitting the data into features and target"""

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

print(X)

print(Y)

"""Split data into train and test data"""

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

"""data stored in x_train and x_test
and lable stored in y_train and y_test"""

print(X.shape, X_train.shape, X_test.shape)

"""model training
logistic regression"""

model = LogisticRegression()

#training the logistic regression model with training data
model.fit(X_train,Y_train)

"""model evaluation and accuracy"""
```

CREDIT CARD FRAUD DETECTION

```
#accuracy on training data
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)
>>Accuracy on Training data :  0.9428208386277002

#accuracy on test data
X_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction,Y_test)


print('Accuracy on Testing data : ',testing_data_accuracy)
>>Accuracy on Testing data :  0.9289340101522843
```