1: Credit Card Detection Machine Learning Project with Report:

Project Report:

Introduction

Credit card fraud is a major concern for both consumers and financial institutions. Fraudulent transactions can lead to financial losses and damage to the reputation of financial institutions. Machine learning techniques have been used extensively to detect fraudulent transactions. In this project, we use logistic regression to classify transactions as either legitimate or fraudulent based on their features.

Data

The data used in this project is a CSV file containing credit card transaction data. The data has 31 columns and 284,807 rows. The "Class" column is the target variable, which indicates whether the transaction is legitimate (Class = 0) or fraudulent (Class = 1).

Preprocessing

Before training the model, we first separate the legitimate and fraudulent transactions. Since the data is imbalanced, with significantly more legitimate transactions than fraudulent transactions, we undersample the legitimate transactions to balance the classes. We then split the data into training and testing sets using the train_test_split () function.

Model

We use logistic regression to classify transactions as either legitimate or fraudulent based on their features. Logistic regression is a widely used classification algorithm that models the probability of an event occurring based on input features. The logistic regression model is trained on the training data using the LogisticRegression () function from scikit-learn. The trained model is then used to predict the target variable for the testing data.

Evaluation

The performance of the model is evaluated using the accuracy metric, which is the fraction of correctly classified transactions. The accuracy on the training and testing data is calculated using the accuracy_score() function from scikit-learn.

Streamlit Application

We use Streamlit to create a user interface for the credit card fraud detection project. The Streamlit application allows the user to upload a CSV file containing credit card transaction data, and the uploaded data is used to train the logistic regression model. The user can also input transaction features and get a prediction on whether the transaction is legitimate or fraudulent.

Conclusion

In this project, we used logistic regression to detect fraudulent credit card transactions. We achieved a high accuracy on both the training and testing data, indicating that the model is effective at detecting fraudulent transactions. The Streamlit application provides an easy-to-use interface for detecting fraudulent transactions in real-time.

```
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

credit_card_data = pd.read_csv('creditcard.csv')
credit_card_data.head()

credit_card_data.sample()

# dataset informations
credit_card_data.info()

# checking the number of missing values in each column
credit_card_data.isnull().sum()

# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()

legit = credit_card_data[credit_card_data.Class==0]
fraud = credit_card_data[credit_card_data['Class']==1]
fraud['Class']


# statistical measures of the data
legit.Amount.describe()


fraud.Amount.describe()

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

legit_sample = legit.sample(n=492)

new_df = pd.concat([legit_sample,fraud],axis=0)

new_df

new_df['Class'].value_counts()

new_df.groupby('Class').mean()


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

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


model=LogisticRegression()

# training the Logistic Regression Model with Training Data
model.fit(X_train, Y_train)
# 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 test data
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)

"0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0")
```

About The Code:

This project implements a **machine learning-based Credit Card Fraud Detection System** using **Logistic Regression** and a **Streamlit web interface**.
The goal of the system is to classify transactions as *legitimate* or *fraudulent* based on numerical transaction features.

Code Overview

1. **Library Imports:**
   The project begins by importing essential Python libraries —
   numpy and pandas for data manipulation, scikit-learn for model training and evaluation, and streamlit for building the interactive web application.

2. **Streamlit Page Configuration:**
   The layout and title of the app are defined using st.set_page_config().
   This ensures a clean and wide display layout suitable for data exploration and user inputs.

3. **Data Loading:**
   The load_data() function is responsible for reading a user-uploaded CSV file containing credit card transactions.
   It uses st.cache() to optimize performance by preventing redundant reloads of the same data.

4. **Model Training:**
   The train_model() function handles all machine learning operations:

   - It first separates legitimate (Class = 0) and fraudulent (Class = 1) transactions.
   - Since the dataset is highly imbalanced, it **undersamples** legitimate transactions to create a balanced dataset.
   - The data is then split into **training and testing sets** using train_test_split().

- A **Logistic Regression** model is trained on the training data, and its accuracy is evaluated on both training and test sets using accuracy_score().

5. **Transaction Parsing:**
   The helper function parse_transaction_string() converts a comma-separated string of transaction feature values entered by the user into a structured dictionary, matching the dataset's feature columns.

6. **Streamlit User Interface:**
   The main Streamlit interface includes:
   - A **file uploader** (st.file_uploader) allowing users to upload a CSV dataset dynamically.
   - Display of dataset shape and model performance (training and test accuracies).
   - A **transaction input section**, where users can input numerical transaction values in a comma-separated format.
   - Upon submission, the system predicts whether the entered transaction is **legitimate** or **fraudulent** and displays the result instantly.

7. **Prediction Output:**
   The prediction result is presented clearly using st.success() for legitimate transactions and st.error() for fraudulent ones, providing immediate visual feedback to the user.

Summary

This streamlined version of the project demonstrates how **machine learning** and **Streamlit** can be combined to build an **interactive fraud detection tool**.
The application allows users to:

- Upload and analyze their own transaction datasets,
- Automatically train a logistic regression model,
- View model accuracy results, and
- Test new transactions in real-time.

The modular design (data loading, model training, and user interaction handled separately) makes the code **simple, efficient, and easily extensible** for future improvements such as scaling, feature engineering, or deploying more advanced ML algorithms.