CREDIT CARD FRAUD DETECTION

PHASE 3

AI & ADS:

Data loading and preprocessing phase sets the foundation for building accurate credit card fraud detection models using AI and ADS technologies**.**

DATA LOADING:

We have collected the reliable dataset containing credit card transaction records, with labels indicating whether each transaction is fraudulent or legitimate. This dataset contains the sequence of transaction information.

1. **Import Libraries:**
   Import the required python libraries ,especially Pandas for data manipulation and analysis.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from imblearn.over_sampling import SMOTE
```

2. **Load the Dataset:**
   Load the raw transaction data from your data source into your chosen data analysis tool or programming environment

```
Load your credit card fraud dataset (replace with your actual dataset path)

df = pd.read_csv("credit_card_fraud_dataset.csv")
```

3. **Explore the Dataset:**

Includes initial data inspection ,check for missing values and get basic statistics.

```
print(df.head())

print(df.info())

print(data.isnull().sum())

print(data.describe())

print(df['Class'].value_counts())
```

4. **Split Data into Features and Target**:
   Split the dataset into training and testing sets to evaluate your model's performance. A common split is 70-30 or 80-20.

```
X = df.drop('Class', axis=1)
```

y = df['Class']

DATA PREPROCESSING:

5**.Data cleaning:**

Handling missing values, Data cleaning, Feature selection and Data transformation:

```
data['amt'].fillna(data['amt'].mean(), inplace=True)

data.drop_duplicates(inplace=True)

selected_columns = ['trans_date_trans_time', 'cc_num', 'merchant', 'category', 'amt', 'is_fraud']

data = data[selected_columns]

data['trans_date_trans_time'] = pd.to_datetime(data['trans_date_trans_time'])

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data['amt'] = scaler.fit_transform(data[['amt']])
```

6. Handling Imbalanced Data:

Credit card fraud datasets are often highly imbalanced, with a small number of fraudulent transactions. You can oversample the minority class using techniques like SMOTE (Synthetic Minority Over-sampling Technique):

```
smote = SMOTE(sampling_strategy=0.5)  # Adjust the sampling_strategy as needed

X_resampled, y_resampled = smote.fit_resample(X, y)
```

7. Splitting Data:

Divide the dataset into training and testing sets for model evaluation.

```
from sklearn.model_selection import train_test_split

X = data.drop('is_fraud', axis=1)

y = data['is_fraud']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

PYTHON PROGRAM USING THE RANDOM FOREST CLASSIFIER:

```
# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier
```

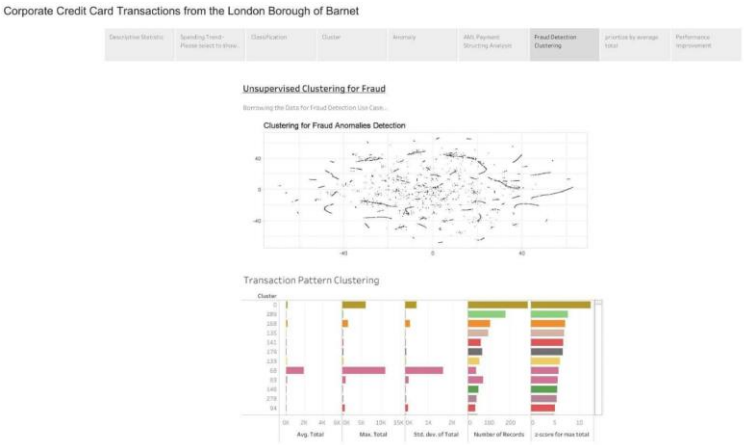
```
from sklearn.metrics import classification_report, accuracy_score

# Load the dataset

data = pd.read_csv('your_dataset.csv')

# Data Preprocessing

# ... (follow the preprocessing steps described in the previous response)

# Handle imbalanced data (oversampling or undersampling)

# Split the data into training and testing sets

X = data.drop('is_fraud', axis=1)

y = data['is_fraud']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Feature Scaling (if not done during preprocessing)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

# Train a Machine Learning Model (Random Forest Classifier as an example)

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)

# Make predictions on the test set

y_pred = model.predict(X_test)

# Model Evaluation

accuracy = accuracy_score(y_test, y_pred)

classification_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)

print("Classification Report:\n", classification_report)
```

DATA ANALYTICS AND VISUALIZATION:

DAC tools primarily focus on data analysis and visualization, they can be used in conjunction with data preprocessing tools to prepare data effectively.

- **Tableau or Power BI**: These are user-friendly, interactive tools for creating insightful dashboards and visualizing patterns in credit card transaction data**.**
- **QlikView or Qlik Sense:** These tools offer associative data models for intuitive data exploration.

- **R and RStudio**: R is a powerful language for statistical analysis and visualization.Python with Matplotlib, Seaborn, and Plotly: These libraries enhance data visualization capabilities in Python.



IOT DEVICES:

IoT devices can play a significant role in credit card fraud detection by providing real-time data and enhancing security.

Deploying IoT devices and developing Python scripts as described in this process creates a real-time credit card fraud detection system that can enhance security at physical points of sale. It allows for quick identification of potential fraud, leading to timely responses to protect cardholders and financial institutions

- Card Readers*: IoT card readers can be deployed at point-of-sale (POS) terminals or ATMs to capture card data securely. These readers are equipped with encryption capabilities and tamper-detection features.
- Remote Locking/Blocking Devices*: IoT devices can be used to remotely lock or block access to a compromised credit card or ATM, adding an extra layer of security.
- Smart Card Technology*: IoT-enabled smart cards have built-in chips and sensors that can monitor card usage and communicate with a central system for fraud detection.
- RFID/NFC Sensors*: RFID (Radio-Frequency Identification) and NFC (Near Field Communication) sensors can be used to track card movements and transactions. They can provide data on when and where cards are used.

Develop Python scripts that run on the IoT devices. These scripts should:

- ✓ Collect transaction data from the card readers and sensors.
- ✓ Format the data for transmission.
- ✓ Securely transmit the data to the central server.

The central server's Python script includes a credit card fraud detection algorithm. This algorithm assesses the incoming transaction data for signs of potential fraud

When a potentially fraudulent transaction is detected, the central server's script sends real-time alerts and notifications to relevant parties, such as security teams or financial institutions. These alerts prompt immediate action.