Import numpy as np

Import pandas as pd

From sklearn.model\_selection import train\_test\_split

From sklearn.ensemble import RandomForestClassifier

From sklearn.metrics import classification\_report, confusion\_matrix

# 1. Simulate Transaction Data

Def generate\_transaction(account\_id, is\_fraud=False):

Amount = np.random.uniform(10, 500)

Transaction\_time = pd.to\_datetime(‘now’)

Location = np.random.choice([‘Online’, ‘Physical Store’])

Ip\_address = f”192.168.1.{np.random.randint(1, 255)}” if location == ‘Online’ else None

Device = np.random.choice([‘Mobile’, ‘Desktop’, ‘Tablet’]) if location == ‘Online’ else None

Transaction = {

‘account\_id’: account\_id,

‘amount’: amount,

‘transaction\_time’: transaction\_time,

‘location’: location,

‘ip\_address’: ip\_address,

‘device’: device,

‘is\_fraud’: is\_fraud

}

Return transaction

Def generate\_synthetic\_data(num\_samples=1000, fraud\_ratio=0.05):

Data = []

Num\_fraud = int(num\_samples \* fraud\_ratio)

Num\_genuine = num\_samples – num\_fraud

For i in range(num\_genuine):

Data.append(generate\_transaction(f”user\_{i}”))

For i in range(num\_fraud):

Data.append(generate\_transaction(f”fraudster\_{i}”, is\_fraud=True))

Return pd.DataFrame(data)

# 2. Load and Preprocess Data

Data = generate\_synthetic\_data(num\_samples=2000, fraud\_ratio=0.03)

# Simple Feature Engineering

Data[‘transaction\_hour’] = data[‘transaction\_time’].dt.hour

Data[‘is\_online’] = data[‘location’].apply(lambda x: 1 if x == ‘Online’ else 0)

Data = pd.get\_dummies(data, columns=[‘location’, ‘device’], drop\_first=True)

Data = data.drop([‘account\_id’, ‘transaction\_time’, ‘ip\_address’], axis=1, errors=’ignore’)

# Separate features (X) and target (y)

X = data.drop(‘is\_fraud’, axis=1)

Y = data[‘is\_fraud’]

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 3. Train a Fraud Detection Model (Random Forest as an example)

Model = RandomForestClassifier(random\_state=42)

Model.fit(X\_train, y\_train)

# 4. Fraud Scoring and Prediction

Def predict\_fraud(transaction\_data, model):

Transaction\_df = pd.DataFrame([transaction\_data])

# Preprocess the new transaction data consistently with the training data

Transaction\_df[‘transaction\_hour’] = transaction\_df[‘transaction\_time’].dt.hour

Transaction\_df[‘is\_online’] = transaction\_df[‘location’].apply(lambda x: 1 if x == ‘Online’ else 0)

Transaction\_df = pd.get\_dummies(transaction\_df, columns=[‘location’, ‘device’], drop\_first=True)

Transaction\_df = transaction\_df.drop([‘account\_id’, ‘transaction\_time’, ‘ip\_address’], axis=1, errors=’ignore’)

# Ensure the new data has the same columns as the training data

Missing\_cols = set(X\_train.columns) – set(transaction\_df.columns)

For c in missing\_cols:

Transaction\_df[c] = 0

Transaction\_df = transaction\_df[X\_train.columns] # Ensure correct column order

Probability = model.predict\_proba(transaction\_df)[:, 1]

Prediction = model.predict(transaction\_df)[0]

Return probability[0], prediction

# 5. Transaction Guarding Function

Def guard\_transaction(transaction):

Fraud\_probability, is\_fraud = predict\_fraud(transaction, model)

Print(f”Transaction Details: {transaction}”)

Print(f”Fraud Probability: {fraud\_probability:.4f}”)

If is\_fraud == 1:

Print(“Potential Fraud Detected! Transaction blocked.”)

Return “Blocked”

Else:

Print(“Transaction Approved.”)

Return “Approved”

# 6. Evaluate the Model

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

Print(“\n--- Model Evaluation ---“)

Print(“Confusion Matrix:\n”, confusion\_matrix(y\_test, y\_pred))

Print(“\nClassification Report:\n”, classification\_report(y\_test, y\_pred))

# 7. Simulate a New Transaction and Guard It

New\_transaction = generate\_transaction(“new\_user”, is\_fraud=False)

Guard\_transaction(new\_transaction)

Suspicious\_transaction = {

‘account\_id’: ‘suspicious\_user’,

‘amount’: 450,

‘transaction\_time’: pd.to\_datetime(‘now’),

‘location’: ‘Online’,

‘ip\_address’: ’10.0.0.1’,

‘device’: ‘Unknown’

}

Guard\_transaction(suspicious\_transaction)