EX.NO.: 04
DATE: 04.07.2025

FRAUDULENT ECOMMERCE TRANSACTION

AIM
To build a machine learning-based Intrusion Detection System (IDS) that detects fraudulent e-commerce transactions by analyzing patterns in customer behavior, transaction attributes, and device/location metadata

ALGORITHM
1. Load the e-commerce transactions dataset into a DataFrame.
2. Remove irrelevant or high-cardinality columns like IDs and addresses.
3. Encode all categorical columns using Label Encoding.
4. Separate the features (X) and target label (`Is Fraudulent`).
5. Normalize numerical features using StandardScaler.
6. Split the data into training and testing sets (e.g., 80:20 ratio).
7. Train a Random Forest Classifier on the training data.
8. Evaluate the model using accuracy, precision, recall, F1-score, and confusion matrix.
9. Create a function to predict fraud status for a new transaction.
10. Use the trained model, encoders, and scaler to process and classify new transactions.

CODE AND OUTPUT
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import joblib

# Load the dataset
df = pd.read_csv("ecommerce_fraud_cleaned1.csv")  # Replace with your actual file name

# Drop columns not needed for prediction (identifiers and high-cardinality text fields)
df = df.drop(['Transaction ID', 'Customer ID', 'IP Address', 'Shipping Address',
'Billing Address', 'Transaction Date'], axis=1)

# Label encode categorical features
categorical_cols = ['Payment Method', 'Product Category', 'Customer Location', 'Device
Used']
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Separate features and label
X = df.drop('IsFraudulent', axis=1)
y = df['IsFraudulent']

# Scale numeric features
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Save model and scaler
joblib.dump(model, "fraud_detection_model.pkl")
joblib.dump(scaler, "scaler.pkl")
joblib.dump(label_encoders, "label_encoders.pkl")

# ----------- Prediction Function -----------
def predict_transaction(transaction_dict):
    input_df = pd.DataFrame([transaction_dict])

    # Encode categorical features safely
    for col in categorical_cols:
        le = label_encoders[col]
        value = input_df[col].iloc[0]
        if value in le.classes_:
            input_df[col] = le.transform([value])
        else:
            # Handle unseen label — assign a default or most frequent category (here,
index 0)
            print(f"[WARNING] Unseen label '{value}' in column '{col}'. Assigning
default.")
            input_df[col] = [0]  # fallback to the first known class

    # Drop unused fields (if they exist)
    input_df = input_df.drop(['Transaction ID', 'Customer ID', 'IP Address', 'Shipping
Address', 'Billing Address', 'Transaction Date'], axis=1, errors='ignore')

    # Scale
    input_scaled = scaler.transform(input_df)

    # Predict
    prediction = model.predict(input_scaled)
    return "Fraudulent" if prediction[0] == 1 else "Legitimate"
```

```python
# Example input transaction
new_transaction = {
    'Transaction ID': 'TX12345',
    'Customer ID': 'C789',
    'Transaction Amount': 500.00,
    'Transaction Date': '2025-07-04 14:32:00',
    'Payment Method': 'Credit Card',
    'Product Category': 'Electronics',
    'Quantity': 2,
    'Customer Age': 28,
    'Customer Location': 'New York',
    'Device Used': 'Mobile',
    'IP Address': '192.168.1.1',
    'Shipping Address': '123 Main St, NY',
    'Billing Address': '123 Main St, NY',
    'Account Age Days': 120,
    'Transaction Hour': 14
}

print("\nPrediction for new transaction:", predict_transaction(new_transaction))
```

```
Confusion Matrix:
 [[13270    49]
 [  560   121]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     13319
           1       0.71      0.18      0.28       681

    accuracy                           0.96     14000
   macro avg       0.84      0.59      0.63     14000
weighted avg       0.95      0.96      0.94     14000

[WARNING] Unseen label 'Credit Card' in column 'Payment Method'. Assigning default.
[WARNING] Unseen label 'Electronics' in column 'Product Category'. Assigning default.
[WARNING] Unseen label 'New York' in column 'Customer Location'. Assigning default.
[WARNING] Unseen label 'Mobile' in column 'Device Used'. Assigning default.

Prediction for new transaction: Legitimate
```

INFERENCE

The Random Forest-based fraud detection system accurately distinguishes between legitimate and fraudulent transactions by learning from historical patterns in payment methods, device types, locations, transaction amounts, and customer behavior. It can generalize to new transactions with high reliability when properly encoded and scaled.