/*Practical No : 06,
Name: Dhiraj Vijay Barwal ,
Roll No : 08,
Class: TECSD,
Batch: T1 */

Predict the credit worthiness of customer credit card fraud detection.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler

# Load the dataset
file_path = '/mnt/data/creditfraud.csv'
df = pd.read_csv(file_path)

# Preprocessing
df = df.drop(columns=['TransactionID', 'TransactionDate'])
df = pd.get_dummies(df, columns=['TransactionType', 'Location'], drop_first=True)

# Define features (X) and target (y)
X = df.drop(columns=['IsFraud'])
y = df['IsFraud']

# Split the 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, stratify=y
)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
model.fit(X_train, y_train)

# Predict the labels for the test data
y_pred = model.predict(X_test)

# Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_rep)
print("\nConfusion Matrix:\n", conf_matrix)

# Predict fraud status for a new transaction
new_transaction = [[2500, 503, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0]]  # Adjusted to match training data
features
new_transaction_scaled = scaler.transform(new_transaction)
fraud_prediction = model.predict(new_transaction_scaled)
print(f"\nThe transaction is predicted as: {'Fraud' if fraud_prediction[0] == 1 else 'Legitimate'}")
```

```python
# Preprocessing
df = df.drop(columns=['TransactionID', 'TransactionDate'])
df = pd.get_dummies(df, columns=['TransactionType', 'Location'], drop_first=True)

# Define features (X) and target (y)
X = df.drop(columns=['IsFraud'])
y = df['IsFraud']

# Split the 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, stratify=y
)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
model.fit(X_train, y_train)

# Predict the labels for the test data
y_pred = model.predict(X_test)

# Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("\nClassification Report:\n", classification_rep)
print("\nConfusion Matrix:\n", conf_matrix)

# Predict fraud status for a new transaction
new_transaction = [[2500, 503, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0]]  # Adjusted to match training
    data features
new_transaction_scaled = scaler.transform(new_transaction)
fraud_prediction = model.predict(new_transaction_scaled)
print(f"\nThe transaction is predicted as: {'Fraud' if fraud_prediction[0] == 1 else
    'Legitimate'}")
```

```
> Accuracy: 98.99%

Classification Report:
                precision    recall  f1-score   support

           0       0.99      1.00      0.99     19800
           1       0.00      0.00      0.00       200

    accuracy                           0.99     20000
   macro avg       0.49      0.50      0.50     20000
weighted avg       0.98      0.99      0.98     20000

Confusion Matrix:
[[19798     2]
 [  200     0]]

The transaction is predicted as: Legitimate
```