```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE

# Step 1: Create simple transaction dataset
data = {
    'amount': [100, 9500, 50, 6000, 200, 7500],
    'hour': [12, 1, 15, 3, 14, 23],
    'type': ['online', 'in_store', 'in_store', 'online', 'in_store',
    'is_fraud': [0, 1, 0, 1, 0, 1]
}
df = pd.DataFrame(data)

# Step 2: Encode categorical column
df = pd.get_dummies(df, columns=['type'], drop_first=True)

# Step 3: Features and target
X = df.drop('is_fraud', axis=1)
y = df['is_fraud']

# Step 4: Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Handle imbalance
X_resampled, y_resampled = SMOTE().fit_resample(X_scaled, y)

# Step 6: Train model
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_re
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 7: Evaluate
y_pred = model.predict(X_test)
print("\nClassification Report:\n", classification_report(y_test, y_p

# Step 8: Simple fraud guard function
def guard(transaction):
    tx = pd.DataFrame([transaction])
    tx = pd.get_dummies(tx)
    for col in X.columns:
        if col not in tx.columns:
            tx[col] = 0
    tx = tx[X.columns]
    tx_scaled = scaler.transform(tx)
    return "Fraud" if model.predict(tx_scaled)[0] == 1 else "Safe"

# Step 9: Test a transaction
test_txn = {'amount': 8000, 'hour': 2, 'type': 'online'}
print("Transaction status:", guard(test_txn))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
```

ValueError TypeError C


```
import pandas as pd
from sklearn.ensemble import RandomFor
from sklearn.model_selection import tr
from sklearn.preprocessing import Star
from sklearn.metrics import classifica
from imblearn.over_sampling import SMO

# Step 1: Create simple transaction da
data = {
    'amount': [100, 9500, 50, 6000, 20
    'hour': [12, 1, 15, 3, 14, 23],
    'type': ['online', 'in_store', 'ir
    'is_fraud': [0, 1, 0, 1, 0, 1]
}
df = pd.DataFrame(data)

# Step 2: Encode categorical column
df = pd.get_dummies(df, columns=['type

# Step 3: Features and target
X = df.drop('is_fraud', axis=1)
y = df['is_fraud']

# Step 4: Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Handle imbalance
X_resampled, y_resampled = SMOTE().fit

# Step 6: Train model
X_train, X_test, y_train, y_test = tra
model = RandomForestClassifier(random_
model.fit(X_train, y_train)

# Step 7: Evaluate
y_pred = model.predict(X_test)
print("\nClassification Report:\n", cl

# Step 8: Simple fraud guard function
def guard(transaction):
    tx = pd.DataFrame([transaction])
    tx = pd.get_dummies(tx)
    for col in X.columns:
        if col not in tx.columns:
            tx[col] = 0
    tx = tx[X.columns]
    tx_scaled = scaler.transform(tx)
    return "Fraud" if model.predict(tx

# Step 9: Test a transaction
test_txn = {'amount': 8000, 'hour': 2,
print("Transaction status:", guard(tes
```

```
                1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

Transaction status: Fraud
```

```
Classification Report:
              precision    recall  f1-

           0       1.00      1.00
           1       1.00      1.00

    accuracy
   macro avg       1.00      1.00
weighted avg       1.00      1.00

Transaction status: Fraud
```