MAIN PROJECT: Online Payments Fraud Detection with Machine Learning

Dataset:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	\
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	

	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	M1979787155	0.0	0.0	0	0
1	M2044282225	0.0	0.0	0	0
2	C553264065	0.0	0.0	1	0
3	C38997010	21182.0	0.0	1	0
4	M1230701703	0.0	0.0	0	0

Project Code:

```
import pandas as pd
from imblearn.combine import SMOTETomek
from imblearn.over_sampling import SMOTE
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.utils.class_weight import compute_class_weight
import numpy as np
df = pd.read_csv('dataset.csv')
df = df.drop(columns=['nameOrig', 'nameDest', 'Unnamed: 11'])
df = pd.get_dummies(df, columns=['type'], drop_first=True)
df['balance_change'] = df['oldbalanceOrg'] - df['newbalanceOrig']
df['amount_ratio'] = df['amount'] / (df['oldbalanceOrg'] + 1e-5)
imputer = SimpleImputer(strategy='mean')
```

```
X = df.drop(columns=['isFraud'])
X_imputed = imputer.fit_transform(X)
y = df['isFraud']
smote = SMOTE(sampling_strategy='auto', k_neighbors=1, random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_imputed, y)
smote_tomek = SMOTETomek(smote=smote, random_state=42)
X_resampled, y_resampled = smote_tomek.fit_resample(X_resampled, y_resampled)
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3,
random_state=42)
class_weights = compute_class_weight('balanced', classes=np.array([0, 1]), y=y_resampled)
clf = RandomForestClassifier(class_weight={0: class_weights[0], 1: class_weights[1]})
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_pred_prob = clf.predict_proba(X_test)[:, 1]
print(classification_report(y_test, y_pred, zero_division=1))
print(f'AUC: {roc_auc_score(y_test, y_pred_prob)}')
```

Output:

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           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
```