Online Payments Fraud Detection with Machine Learning

For the major project, I developed a machine learning model for Online Payment Fraud Detection. This involved classifying transactions as either fraudulent or non-fraudulent. I worked on data preprocessing, model training, and evaluating the model's performance to ensure it effectively detects fraud. The project, including all relevant code and findings, was submitted in PDF format as per the instructions in the attached document.

The dataset consists of 10 variables:

- **step**: represents a unit of time where 1 step equals 1 hour
- **type**: type of online transaction
- **amount**: the amount of the transaction
- **nameOrig**: customer starting the transaction
- **oldbalanceOrg**: balance before the transaction
- **newbalanceOrig**: balance after the transaction
- **nameDest**: recipient of the transaction
- **oldbalanceDest**: initial balance of recipient before the transaction
- **newbalanceDest**: the new balance of recipient after the transaction
- **isFraud**: fraud transaction
-

1. Import necessary libraries

- We import necessary libraries for data manipulation, machine learning, and evaluation.

```
import pandas as pd
import numpy as np
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, accuracy_score
```

2. Load the data

- We load the dataset using pandas.

```
data = pd.read_csv('given_dataset.csv')

# Update the path to your dataset

data.head()

# Display the first few rows of the dataframe
```

```
   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          181.6            0.00
3     1  CASH OUT    181.00   C840083671          181.0            0.00
4     1   PAYMENT  11668.14  C2048537726        41554.0        29885.86

      nameDest  oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud
0  M1979787155               0               0        0               0
1  M2044282225               0               0        0               0
2   C553264665               0               0        1               0
3    C38997610           21182               0        1               0
4  M1230701703               0               0        0               0
```

3. Preprocessing

- Encoding - We use LabelEncoder to convert categorical columns (type, nameOrig, nameDest) into numeric values.

```
# Convert categorical features to numerical

label_encoders = {}
for column in ['type', 'nameOrig', 'nameDest']:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le
```

- Feature Scaling - We scale features using StandardScaler to standardize the feature values.

```
# Feature selection and extraction

features = ['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg',
'newbalanceOrig',
            'nameDest', 'oldbalanceDest', 'newbalanceDest']
X = data[features]
y = data['isFraud']


# Scale features

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

4. Train-Test Split

- We split the dataset into training and testing subsets to evaluate the model's performance on unseen data.

```
# Split the data 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)
```

5. Model Training

- We train a RandomForestClassifier to classify transactions as fraudulent or non-fraudulent.

```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
# Make predictions
y_pred = clf.predict(X_test)
```

6. Model Evaluation

- We evaluate the model's performance using accuracy and a classification report, which includes precision, recall, and F1-score.
-

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1
```

7. Conclusion

In this notebook, we implemented a basic machine learning pipeline for classifying fraudulent online payments using a given dataset. The key steps included:

- Data Preprocessing: We transformed categorical features into numerical values and standardized the feature scales to ensure consistent input for the model.
- Model Training: We used a RandomForestClassifier to train the model on the preprocessed data.
- Evaluation: We assessed the model's performance using accuracy and a classification report, providing insights into how well the model identifies fraudulent transactions.

The RandomForestClassifier showed how machine learning can be applied to detect fraudulent transactions based on historical data.