OnlinePymtFraud

February 20, 2025

1 Online Payments Fraud Detection Machine Learning

[2]:
```
# Load Libraries
import pandas as pd
import numpy as np
```

1.1 Load Dataset

[3]:
```
# Kaggle dataset Online Payments Fraud Detection
df = pd.read_csv("/content/onlinefraud.csv")
```

[4]:
```
# Number of Rows and Columns
df.shape
```

[4]:
```
(6362620, 11)
```

[5]:
```
# Display first 5 rows
df.head()
```

[5]:
```
   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
```

[6]:
```
# List Columns and types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

1.2 Data Preparation

[7]:
```
# Checking values for isFlaggedFraud
df.isFlaggedFraud.value_counts()
```

[7]:
```
isFlaggedFraud
0    6362604
1         16
Name: count, dtype: int64
```

[8]:
```
# Checking for nulls
df.isnull().sum()
```

[8]:
```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

[9]:
```
# checking values for type
df.type.value_counts()
```

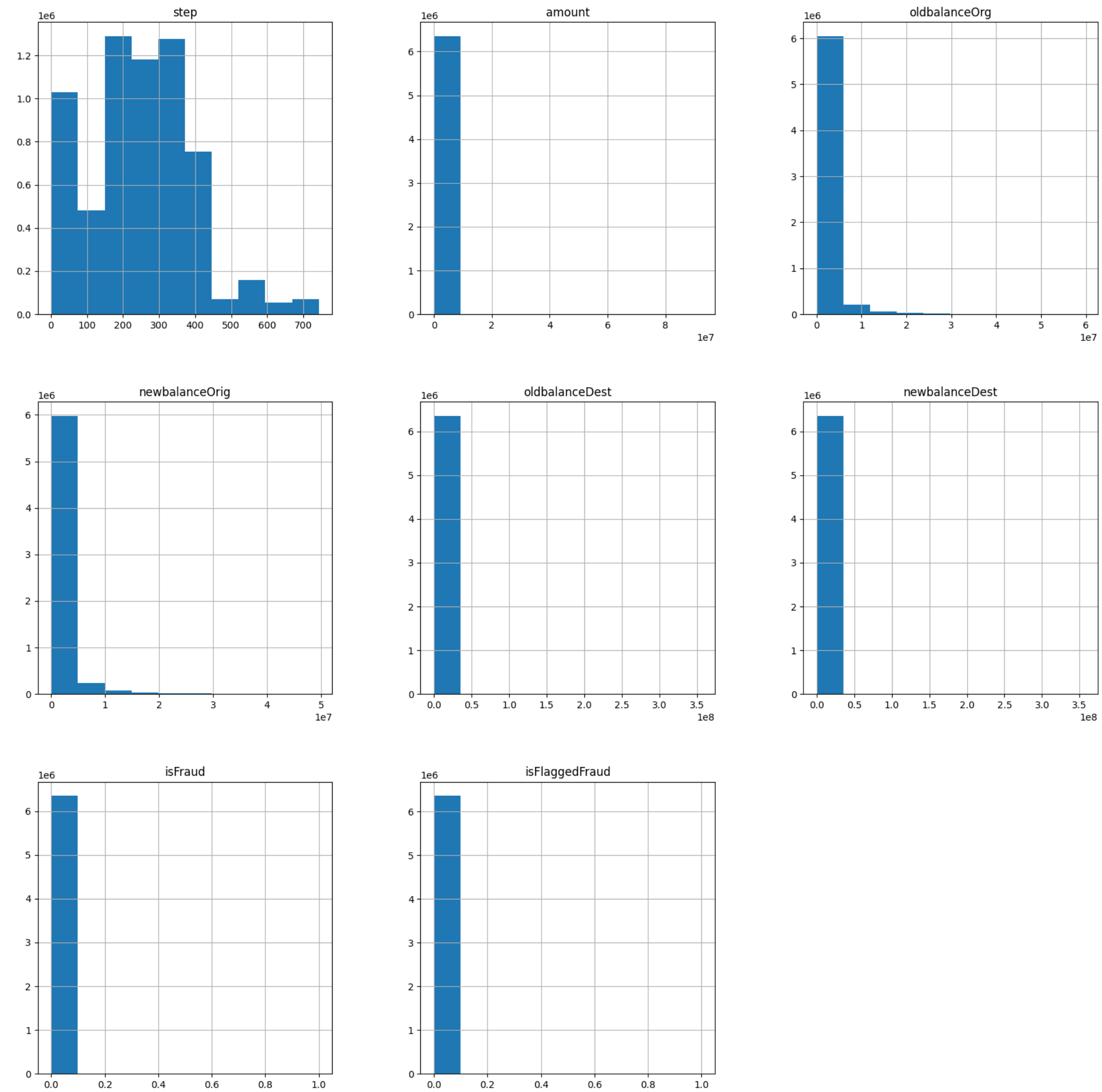
[9]:
```
type
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: count, dtype: int64
```

1.3 Data Visualization

[10]:
```python
# Visualize Categories for Transaction Type
type = df["type"].value_counts()
transactions = type.index
quantity = type.values

import plotly.express as px
figure = px.pie(df,
                values=quantity,
                names = transactions, hole=0.5,
                title = "Distribution of Transaction Type")
figure.show()
```

[11]:
```python
# Histogram for numeric values
import matplotlib.pyplot as plt
df.hist(figsize=(20,20))
plt.show()
```

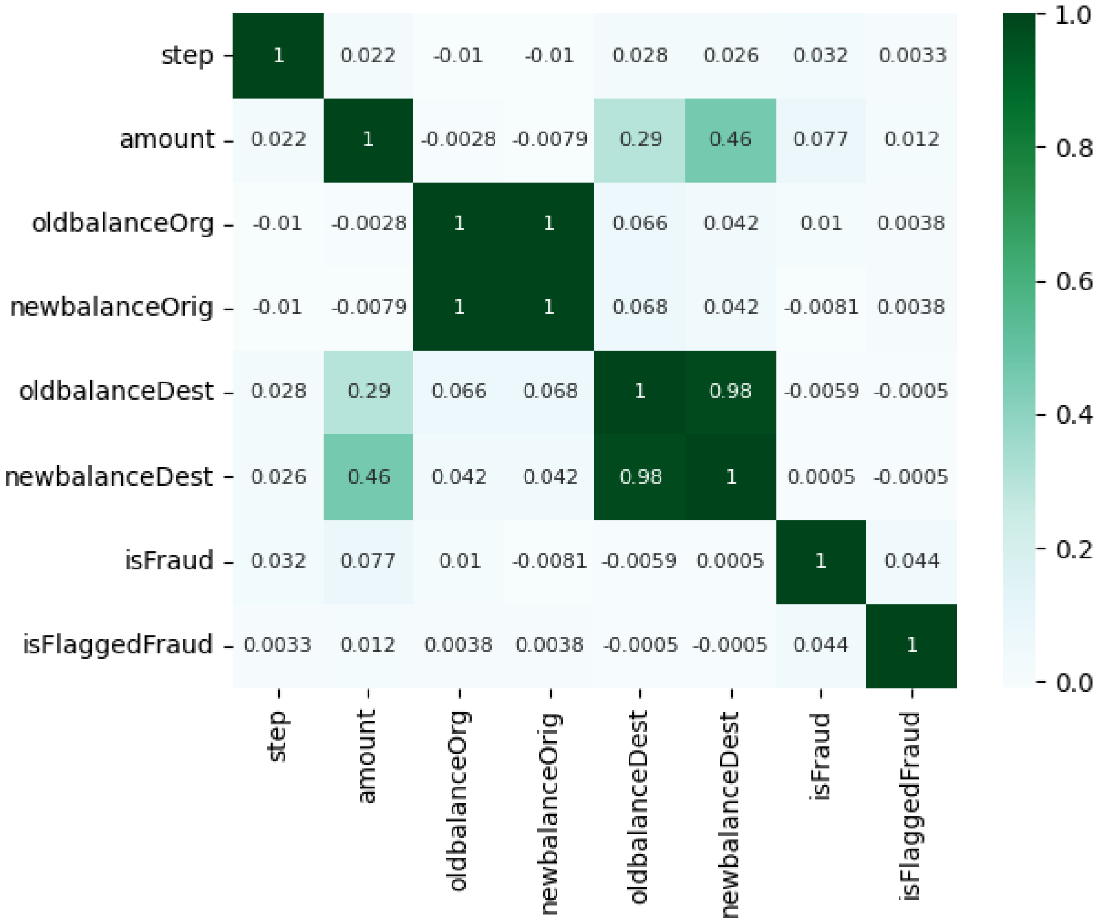


[12]:

```
# Correlation between features and target
correlation = df.corr(numeric_only=True)
print(correlation["isFraud"].sort_values(ascending=False))
```

```
isFraud           1.000000
amount            0.076688
isFlaggedFraud    0.044109
step              0.031578
oldbalanceOrg     0.010154
newbalanceDest    0.000535
oldbalanceDest   -0.005885
newbalanceOrig   -0.008148
Name: isFraud, dtype: float64
```

[13]:
```
# Visualize Correlation
import seaborn as sns
sns.heatmap(round(df.corr(numeric_only=True),4), annot=True,␣
↪cmap="BuGn",annot_kws={'size':8})
plt.show()
```



1.4 Encoding and Correlation

[14]:
```
# Encode categorical feature
df= pd.get_dummies(df,columns=['type'],prefix=['type'],dtype=int)
```

[15]:
```
# Display rows after encoding
df.head()
```

[15]:

	step	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	\
0	1	9839.64	C1231006815	170136.0	160296.36	M1979787155	
1	1	1864.28	C1666544295	21249.0	19384.72	M2044282225	

```
2     1     181.00  C1305486145          181.0            0.00   C553264065
3     1     181.00   C840083671          181.0            0.00    C38997010
4     1   11668.14  C2048537720        41554.0        29885.86  M1230701703

   oldbalanceDest  newbalanceDest  isFraud  isFlaggedFraud  type_CASH_IN  \
0             0.0             0.0        0               0             0
1             0.0             0.0        0               0             0
2             0.0             0.0        1               0             0
3         21182.0             0.0        1               0             0
4             0.0             0.0        0               0             0

   type_CASH_OUT  type_DEBIT  type_PAYMENT  type_TRANSFER
0              0           0             1              0
1              0           0             1              0
2              0           0             0              1
3              1           0             0              0
4              0           0             1              0
```

[16]:

```python
# Drop columns that are not needed
df = df.drop(["step","nameOrig", "nameDest", "oldbalanceDest",␣
 ↪"newbalanceDest", "isFlaggedFraud"], axis=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 9 columns):
 #   Column          Dtype
---  ------          -----
 0   amount          float64
 1   oldbalanceOrg   float64
 2   newbalanceOrig  float64
 3   isFraud         int64
 4   type_CASH_IN    int64
 5   type_CASH_OUT   int64
 6   type_DEBIT      int64
 7   type_PAYMENT    int64
 8   type_TRANSFER   int64
dtypes: float64(3), int64(6)
memory usage: 436.9 MB
```

1.5 Model Building

[17]:

```python
# Machine Learning Libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
```

1.5.1 Split Dataset into Features and Target

[18]:
```
# Split in features (X) and target (y)
X = df.drop("isFraud",axis=1)

print(X.shape)
```

```
(6362620, 8)
```

[19]:
```
y = df['isFraud']
print(y.shape)
```

```
(6362620,)
```

[20]:
```
y.value_counts()
```

[20]:
```
isFraud
0    6354407
1       8213
Name: count, dtype: int64
```

[21]:
```
# Checking values for isFraud
df.tail(10)
```

[21]:

	amount	oldbalanceOrg	newbalanceOrig	isFraud	type_CASH_IN
6362610	63416.99	63416.99	0.0	1	0
6362611	63416.99	63416.99	0.0	1	0
6362612	1258818.82	1258818.82	0.0	1	0
6362613	1258818.82	1258818.82	0.0	1	0
6362614	339682.13	339682.13	0.0	1	0
6362615	339682.13	339682.13	0.0	1	0
6362616	6311409.28	6311409.28	0.0	1	0
6362617	6311409.28	6311409.28	0.0	1	0
6362618	850002.52	850002.52	0.0	1	0
6362619	850002.52	850002.52	0.0	1	0

	type_CASH_OUT	type_DEBIT	type_PAYMENT	type_TRANSFER
6362610	0	0	0	1
6362611	1	0	0	0
6362612	0	0	0	1
6362613	1	0	0	0
6362614	0	0	0	1
6362615	1	0	0	0
6362616	0	0	0	1
6362617	1	0	0	0
6362618	0	0	0	1
6362619	1	0	0	0

1.5.2 Imbalance

[46]:
```
# Address the imbalance between Fraud and Not Fraud
from imblearn.under_sampling import RandomUnderSampler
ros = RandomUnderSampler(sampling_strategy=0.4)
X_ros,y_ros = ros.fit_resample(X,y)
```

[28]:
```
y_ros.value_counts()
```

[28]:
```
isFraud
0    20532
1     8213
Name: count, dtype: int64
```

[29]:
```
X_train, X_test, y_train, y_test = train_test_split(X_ros,y_ros, test_size=0.
↪3,random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(20121, 8)
(8624, 8)
(20121,)
(8624,)
```

1.6 Hyperparameter Tuning

[36]:
```
from sklearn.model_selection import GridSearchCV

model = DecisionTreeClassifier()
grid_params = {
'criterion': ['gini', 'entropy'],
'max_depth': [3,5,7,10],
'min_samples_split': range(2,10,1),
'min_samples_leaf': range(2,10,1)
}

grid_search = GridSearchCV(model, grid_params, cv=5, n_jobs = -1, verbose = 1)
grid_result = grid_search.fit(X_train, y_train)
print('Best Score: %s' % grid_result.best_score_)
print('Best Hyperparameters: %s' % grid_result.best_params_)
```

```
Fitting 5 folds for each of 512 candidates, totalling 2560 fits
Best Score: 0.9926941950779792
Best Hyperparameters: {'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 2, 'min_samples_split': 3}
```

1.6.1 Model

[38]:
```
# model
model  = DecisionTreeClassifier(criterion= 'entropy', max_depth= 10,␣
 ↪min_samples_leaf= 2, min_samples_split= 3, random_state=42,␣
 ↪class_weight='balanced')

# fit
model.fit(X_train, y_train)

# predict
y_pred = model.predict(X_test)
```

1.6.2 Accuracy

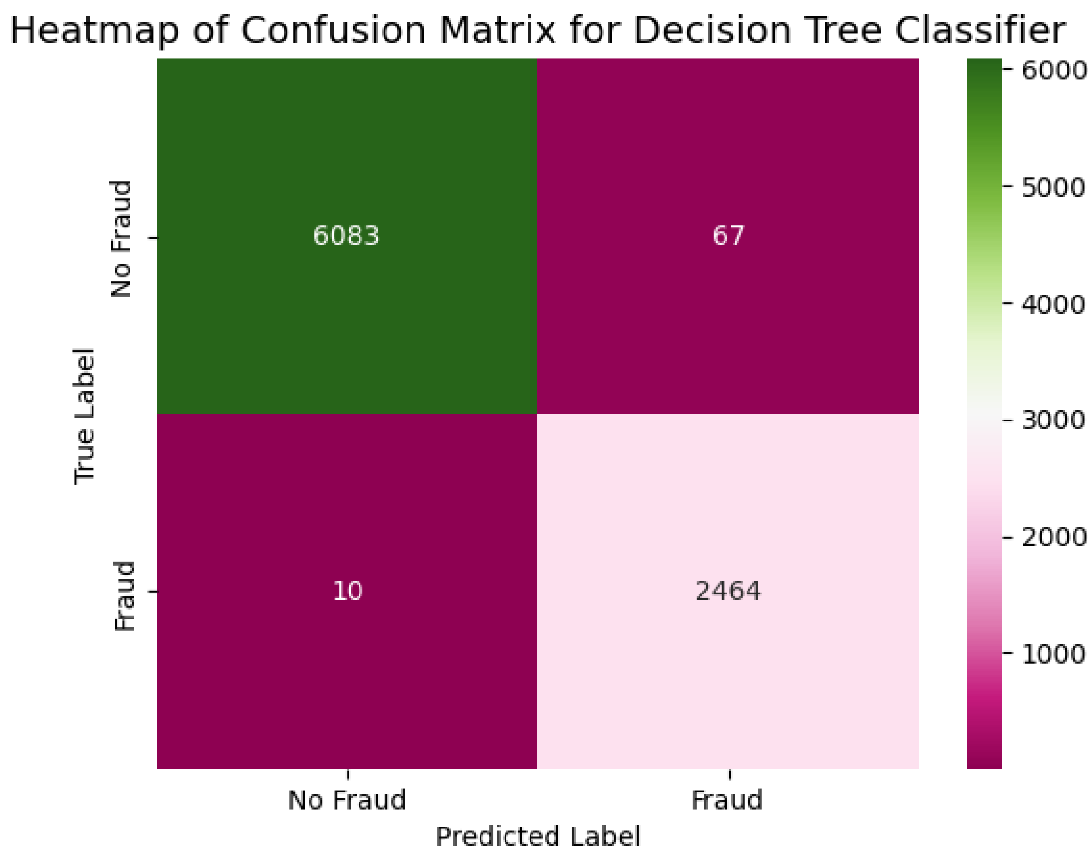
[39]:
```
from sklearn.metrics import classification_report, confusion_matrix

print("Classification Report for Random Forest")
print(classification_report(y_test, y_pred))
classes = ['No Fraud', 'Fraud']
sns.heatmap(confusion_matrix(y_test,y_pred), annot=True,␣
 ↪fmt="d",cmap="PiYG",xticklabels=classes, yticklabels=classes)
plt.title('Heatmap of Confusion Matrix for Decision Tree Classifier', fontsize␣
 ↪= 14)
plt.xlabel('Predicted Label', fontsize = 10) # x-axis label with fontsize 15
plt.ylabel('True Label', fontsize = 10) # y-axis label with fontsize 15
plt.show()
```

```
Classification Report for Random Forest
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      6150
           1       0.97      1.00      0.98      2474

    accuracy                           0.99      8624
   macro avg       0.99      0.99      0.99      8624
weighted avg       0.99      0.99      0.99      8624
```



Model achieved 97% for detecting Fraud

1.7 Predictions

[48]:
```
X.columns
```

[48]:
```
Index(['amount', 'oldbalanceOrg', 'newbalanceOrig', 'type_CASH_IN',
       'type_CASH_OUT', 'type_DEBIT', 'type_PAYMENT', 'type_TRANSFER'],
      dtype='object')
```

[52]:
```
data=[[63416.52,63416.52,0,0,0,0,0,1]]

p = pd.DataFrame(data,columns=['amount', 'oldbalanceOrg', 'newbalanceOrig',␣
 ↪'type_CASH_IN',
       'type_CASH_OUT', 'type_DEBIT', 'type_PAYMENT', 'type_TRANSFER'])
p
```

[52]:
```
     amount  oldbalanceOrg  newbalanceOrig  type_CASH_IN  type_CASH_OUT  \
0  63416.52       63416.52               0             0              0
```

```
   type_DEBIT  type_PAYMENT  type_TRANSFER
0           0             0              1
```

[53]:
```
if model.predict(p) == 0:
  print("Not Fraud")
else:
  print("Fraud")
```

```
Fraud
```

Results: - The online payment data had an imbalance of 6 million rows (No Fraud) to 8213 rows (Fraud) - Handled the imbalance by using RandomUnderSampling to reduce the No Fraud rows.No Fraud rows were reduced to 20532 rows. - Used hyperparameter tuning to determine the best parameters for the Decision Tree model - The Decision Tree model achieved 97% accuracy for detecting Fraud