Online Payment Fraud Detection

Introduction

To identify online payment fraud with machine learning, we need to train a machine learning model for classifying fraudulent and non-fraudulent payments. For this, we need a dataset containing information about online payment fraud, so that we can understand what type of transactions lead to fraud. For this task, I collected a dataset from Kaggle, which contains historical information about fraudulent transactions which can be used to detect fraud in online payments.

Understaing the Dataset

The dataset consists of 11 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

In [2]:

```
# Importing the libraries
#Pandas is a Open source library providing high performance,allows for
import pandas as pd
#Numpy is a core library providing high-performance multidimensional ar
import numpy as np
#Seaborn helps to visualize the data and make it more and more undertak
import seaborn as sns
#Metplotlib is a comprehensive library for creating static,animated,and
import matplotlib.pyplot as plt
#Splitting into random subsets for train and test.
from sklearn.model_selection import train_test_split

```

In [3]:

```
#extracting the dataset.
df=pd.read_csv('D:\\panda\\onlinefraud.csv')
df
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C776
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C2080
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C873

6362620 rows × 11 columns

In [4]:

```
#getting the information of columsns
df.columns
```

Out[4]:

```
Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceO
rig',
       'nameDest', 'oldbalanceDest', 'newbalanceDest', 'isFraud',
       'isFlaggedFraud'],
      dtype='object')
```

In [5]:

```
#displaying the top n lines in the data file.
df.head()
```

Out[5]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703

In [6]:

```
#cleaning of the data
df.drop('isFlaggedFraud', axis=1, inplace=True)

```

In [7]:
```
#displaying the top 10 rows of the data set
df.head(10)
```

Out[7]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703
5	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M573487274
6	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M408069119
7	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333
8	1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M1176932104
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C195600860

EDA (Exploratory data Analysis)

It is an approach that is used to analyze the data and discover trends, patterns, or check assumptions in data with the help of statistical summaries and graphical representations.

In [8]:
```
#getting insights  about  the dataset
#shape of the data(refers the no of rows and columns)
df.shape
```

Out[8]: (6362620, 10)

In [9]:
```
#Let’s get a quick summary of the dataset using the pandas describe() r
df.describe()
```

Out[9]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbala
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561

In [10]:

```
#let’s also see the columns and their data types. For this, we will use
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 10 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  
dtypes: float64(5), int64(2), object(3)
memory usage: 485.4+ MB
```

In [11]:

```
#finding the null values.

df.isnull().sum()
```

Out[11]:

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

There is no missing values.

In [12]:

```
# Exploring transaction type
df['type'].unique()
```

Out[12]:

```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

In [13]:

```
type=df['type'].value_counts()
```

In [14]:

```
transaction=type.index
```

In [15]:
```
type
```

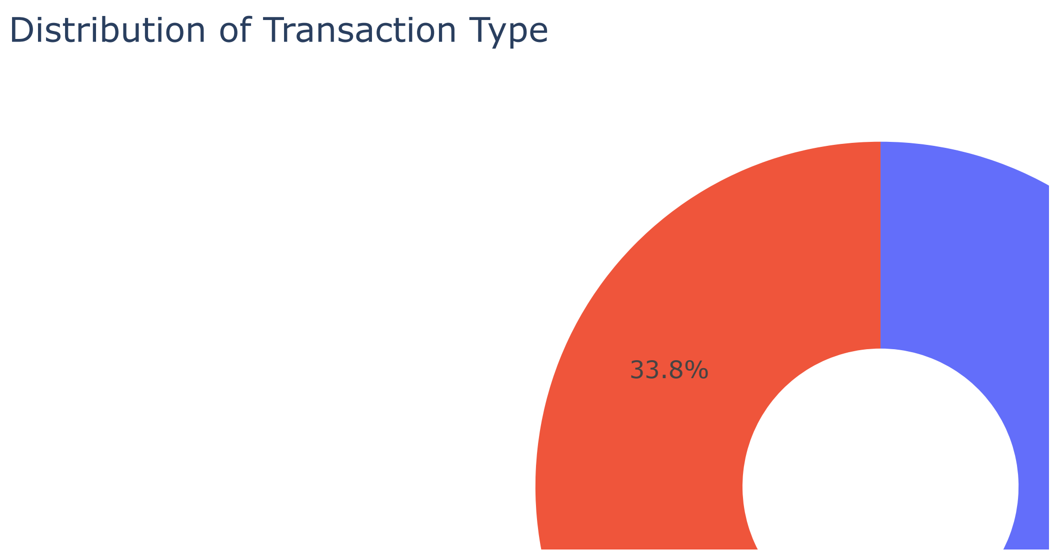
Out[15]:
```
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: type, dtype: int64
```

In [16]:
```
quantity=type.values
```

In [17]:
```
#importing library for plotting charts
import plotly.express as px
```

In [18]:
```
figure = px.pie(df, values=quantity, names=transaction, hole=0.4, titl
```

In [19]:
```
figure.show()
```



In [20]:
```python
#dropping the rows containing the null values.
df.dropna()
```

Out[20]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	na
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M197
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M123
...	...	...	...	...	...	...	
6362615	743	CASH_OUT	339682.13	C786484425	339682.13	0.00	C77
6362616	743	TRANSFER	6311409.28	C1529008245	6311409.28	0.00	C1881
6362617	743	CASH_OUT	6311409.28	C1162922333	6311409.28	0.00	C1365
6362618	743	TRANSFER	850002.52	C1685995037	850002.52	0.00	C208
6362619	743	CASH_OUT	850002.52	C1280323807	850002.52	0.00	C87

6362620 rows × 10 columns

In [21]:
```python
# Now let’s have a look at the correlation between the features of the
# Calculating the relationship between each column in the dataset
correlation = df.corr()
print(correlation["isFraud"].sort_values(ascending=False))
```

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

In [22]:
```python
type
```

Out[22]:
```
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: type, dtype: int64
```

In [23]:
```python
# Now let’s transform the categorical features into numerical.
# Changing CASH_OUT to 1, PAYMENT to 2, CASH_IN to 3, TRANSFER to 4 and
df.replace(to_replace=['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CAS
```

In [24]:

```
df
```

Out[24]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	2	9839.64	C1231006815	170136.00	160296.36	M1979787155
1	1	2	1864.28	C1666544295	21249.00	19384.72	M2044282225
2	1	4	181.00	C1305486145	181.00	0.00	C553264065
3	1	1	181.00	C840083671	181.00	0.00	C38997010
4	1	2	11668.14	C2048537720	41554.00	29885.86	M1230701703
...	...	...	...	...	...	...	...
6362615	743	1	339682.13	C786484425	339682.13	0.00	C776919290
6362616	743	4	6311409.28	C1529008245	6311409.28	0.00	C1881841831
6362617	743	1	6311409.28	C1162922333	6311409.28	0.00	C1365125890
6362618	743	4	850002.52	C1685995037	850002.52	0.00	C2080388513
6362619	743	1	850002.52	C1280323807	850002.52	0.00	C873221189

6362620 rows × 10 columns

In [25]:

```
#Transforming the values of the isFraud column into NoFraud and Fraud
df["isFraud"] = df["isFraud"].map({0: "No Fraud", 1: "Fraud"})
```

In [26]:

```
df
```

Out[26]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	2	9839.64	C1231006815	170136.00	160296.36	M1979787155
1	1	2	1864.28	C1666544295	21249.00	19384.72	M2044282225
2	1	4	181.00	C1305486145	181.00	0.00	C553264065
3	1	1	181.00	C840083671	181.00	0.00	C38997010
4	1	2	11668.14	C2048537720	41554.00	29885.86	M1230701703
...	...	...	...	...	...	...	...
6362615	743	1	339682.13	C786484425	339682.13	0.00	C776919290
6362616	743	4	6311409.28	C1529008245	6311409.28	0.00	C1881841831
6362617	743	1	6311409.28	C1162922333	6311409.28	0.00	C1365125890
6362618	743	4	850002.52	C1685995037	850002.52	0.00	C2080388513
6362619	743	1	850002.52	C1280323807	850002.52	0.00	C873221189

6362620 rows × 10 columns

In [27]:
```
# Splitting the data
x=df[["type", "amount", "oldbalanceOrg", "newbalanceOrig"]]
```

In [28]:
```
y=df.iloc[:,-1]
```

In [29]:
```
y
```

Out[29]:
```
0          No Fraud
1          No Fraud
2             Fraud
3             Fraud
4          No Fraud
             ...
6362615       Fraud
6362616       Fraud
6362617       Fraud
6362618       Fraud
6362619       Fraud
Name: isFraud, Length: 6362620, dtype: object
```

Training a Machine Learning Model (DicisionTreeClassifier)

Decision Tree is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

In [30]:
```
#importing the model
from sklearn.tree import DecisionTreeClassifier

```

In [31]:
```
model = DecisionTreeClassifier()
```

In [32]:
```
#splitting the training and testing data
#here random state controls the shuffling process.
#here RS 42 is the best as it Produce the same results accross a differ
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, ra
```

In [33]:
```
#Applying the model
model.fit(xtrain, ytrain)
```

Out[33]:
```
DecisionTreeClassifier()
```

In [48]:
```
#checking the accuracy
#metrics cn also used to find accuracy or error.
#model.score(xtest, ytest)
print(f"Accuracy rate is {round(model.score(xtest, ytest),5)*100}%")
```

```
Accuracy rate is 99.917%
```

In [35]:
```python
# performing predictions on the test dataset
#features = [type, amount, oldbalanceOrg, newbalanceOrig]
model.predict([[2, 9839.64, 170136.00, 160296.36]])
```

```
C:\Users\Fareen\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng:

X does not have valid feature names, but DecisionTreeClassifier was fitted
with feature names
```

Out[35]:
```
array(['No Fraud'], dtype=object)
```

In [36]:
```python
x
```

Out[36]:

	type	amount	oldbalanceOrg	newbalanceOrig
0	2	9839.64	170136.00	160296.36
1	2	1864.28	21249.00	19384.72
2	4	181.00	181.00	0.00
3	1	181.00	181.00	0.00
4	2	11668.14	41554.00	29885.86
...	...	...	...	...
6362615	1	339682.13	339682.13	0.00
6362616	4	6311409.28	6311409.28	0.00
6362617	1	6311409.28	6311409.28	0.00
6362618	4	850002.52	850002.52	0.00
6362619	1	850002.52	850002.52	0.00

6362620 rows × 4 columns

Applying LogisticRegressionCV

It is a class that implements cross-Validation inside it.This class will train Multiple LogisticRegression models and return the best one. (It cas estimate the Performance of the model with less variance than a single"train-test"set split.

In [37]:
```python
from sklearn.linear_model import LogisticRegressionCV
```

In [38]:
```python
#here k=5 means Splitting the data into k no of folds.
#max_iters Refers to the Maximum no of Iterations taken to converge.
# random state is a lot number of the set generated randomly in any ope
model = LogisticRegressionCV(cv=5, max_iter=500, random_state=0)
```

In [39]:
```
#Applying the model
model.fit(xtrain,ytrain)
```

Out[39]: LogisticRegressionCV(cv=5, max_iter=500, random_state=0)

In [40]:
```
#Checking the Accuracy Rate
#model.score(xtest, ytest)
print(f"Our new Accuracy rate is {round(model.score(xtest, ytest),5)*10
```

```
Our new Accuracy rate is 99.94800000000001%
```

In [41]:
```
# performing predictions on the test dataset
model.predict([[2, 9839.64, 170136.00, 160296.36]])
```

```
C:\Users\Fareen\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng:

X does not have valid feature names, but LogisticRegressionCV was fitted w
ith feature names
```

Out[41]: array(['No Fraud'], dtype=object)

In [42]:
```
model.predict([[1, 8900.2, 8990.2, 0.0]])
```

```
C:\Users\Fareen\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng:

X does not have valid feature names, but LogisticRegressionCV was fitted w
ith feature names
```

Out[42]: array(['Fraud'], dtype=object)

Applying Another Model to improve Accuracy

RandomForestClassifier :It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

In [43]:
```
#Applying the RandomForest Model
from sklearn.ensemble import RandomForestClassifier
```

In [44]:
```
# creating a RF classifier.
#  the n_estimator parameter controls the number of trees inside the cl
#  max_depth governs the maximum height upto which the trees inside the
model = RandomForestClassifier(n_estimators=20, random_state=0, max_dep
```

In [45]:
```
model.fit(xtrain,ytrain)
```

Out[45]: RandomForestClassifier(max_depth=6, n_estimators=20, random_state=0)

In [46]:

```
#checking the Accuracy
model.score(xtest, ytest)
```

Out[46]:

```
0.9991654381371197
```

In [47]:

```
print(f"our new accuracy rate is {round(model.score(xtest, ytest),5)*10
```

```
our new accuracy rate is 99.917%
```

We can conclude that DecisionTreeClassifier model is a pretty good at detecting fraud for this dataset as it is having the high accuracy rate in comparision to other models

In [ ]:

```

```