Transaction Monitoring using ML

- Vikrant Kala

What is Transaction Monitoring?

A process of reviewing customer transactions and matching them against customer risk profile & usual transaction pattern. Subsequently, perform a focused examination of transactions and Identification of Suspicious transactions.

Transactions may include transfers, deposits and withdrawals.

Risk- Based Approach

The system for monitoring and reporting suspicious activity should be risk-based, and should be determined by factors such as the bank's size, the nature of its business, its location, frequency and size of transactions and the types and geographical location of its customers.

Transaction Monitoring Rules

1. Consolidation of transactions close to threshold
2. Overall increase in transaction volume
3. Suspicious spend behavior
4. High Volume of transactions
5. Change in Customer profile before large transactions
6. Payment made with same IP address
7. Frequency of withdrawal and deposits
8. Low buyer diversity
9. High transactions count from new Users

Key Red Flags

- Customer, who is a public official, opens account in the name of a family member who begins making large deposits not consistent with the known sources of legitimate family income.

- Transactions that involve depositing large amounts of cash inconsistent with the normal and expected activity of the customer.
- Customer, who is a student, uncharacteristically transfers or exchanges large sums of money.
- Account shows high velocity in the movement of funds, but maintains low beginning and ending daily balances.
- Transaction involves offshore institutions whose names resemble those of well-known legitimate financial institutions.

Goals:

- Exploratory analysis of data to extract the pattern of fraudulent activities.
- Build a machine learning model to classify fraud and non-fraud transactions.
- Reduce the false negatives by tuning the model.

Objective:

The objective of this notebook is to find the patterns of transactions performed and help algorithms learn those patterns in identifying the fradulent transactions and flag them. The algorithm used for implication of this is **Random Forest Algorithm.**

Dataset:

I referenced Kaggle for the best data set available.

https://www.kaggle.com/c/ieee-fraud-detection/data

The data comes from Vesta's real-world e-commerce transactions and contains a wide range of features from device type to product features.

In [10]:

```
# Vikrant Kala

import pandas as pd
from sklearn import preprocessing

# Loading, Cleaning and Displaying data
# Loading .csv file
df1 = pd.read_csv("train_identity.csv")

# Replacing NaN with default values
df1 = df1.fillna(value = -999)

# Dropping columns
df1.drop(["id_23", "id_27"], axis = 1).values

# Replacing values
df1 = df1.replace("Found", 1)
df1 = df1.replace("NotFound", 2)
df1 = df1.replace("New", 3)
df1 = df1.replace("mobile", 4)
df1 = df1.replace("desktop", 5)
df1 = df1.replace("T", 6)
df1 = df1.replace("F", 7)

#df1.select_dtypes("number")
#df1.select_dtypes("object")

# Typecasting
df1["DeviceInfo"] = df1["DeviceInfo"].astype(str)
df1["id_30"] = df1["id_30"].astype(str)
df1["id_31"] = df1["id_31"].astype(str)
df1["id_33"] = df1["id_33"].astype(str)
df1["id_34"] = df1["id_34"].astype(str)
df1["id_15"] = df1["id_15"].astype(str)
df1["id_23"] = df1["id_23"].astype(str)

# Initializing Encoder
number = preprocessing.LabelEncoder()

# Encoding
df1["DeviceInfo"] = number.fit_transform(df1["DeviceInfo"])
df1["id_30"] = number.fit_transform(df1["id_30"])
df1["id_31"] = number.fit_transform(df1["id_31"])
df1["id_33"] = number.fit_transform(df1["id_33"])
df1["id_34"] = number.fit_transform(df1["id_34"])
df1["id_15"] = number.fit_transform(df1["id_15"])
df1["id_23"] = number.fit_transform(df1["id_23"])

df1.head()
```

Out[10]:

	TransactionID	id_01	id_02	id_03	id_04	id_05	id_06	id_07	id_08	id_09	...	id_
0	2987004	0.0	70787.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	1
1	2987008	-5.0	98945.0	-999.0	-999.0	0.0	-5.0	-999.0	-999.0	-999.0	...	
2	2987010	-5.0	191631.0	0.0	0.0	0.0	0.0	-999.0	-999.0	0.0	...	
3	2987011	-5.0	221832.0	-999.0	-999.0	0.0	-6.0	-999.0	-999.0	-999.0	...	
4	2987016	0.0	7460.0	0.0	0.0	1.0	0.0	-999.0	-999.0	0.0	...	

5 rows × 41 columns

In [11]:

```
df1.shape
```

Out[11]:

```
(144233, 41)
```

In [12]:

```
df1 = df1.iloc[:100000]
df1.shape
```

Out[12]:

```
(100000, 41)
```

In [13]:

```
# Loading .csv file
df2 = pd.read_csv("train_transaction.csv")

# Replacing NaN with default values
df2 = df2.fillna(value = -999)

#df2.select_dtypes("number")
#df2.select_dtypes("object")

# Typecasting
df2["card4"] = df2["card4"].astype(str)
df2["card6"] = df2["card6"].astype(str)
df2["P_emaildomain"] = df2["P_emaildomain"].astype(str)
df2["R_emaildomain"] = df2["R_emaildomain"].astype(str)
df2["M1"] = df2["M1"].astype(str)
df2["M2"] = df2["M2"].astype(str)
df2["M3"] = df2["M3"].astype(str)
df2["M4"] = df2["M4"].astype(str)
df2["M5"] = df2["M5"].astype(str)
df2["M6"] = df2["M6"].astype(str)
df2["M7"] = df2["M7"].astype(str)
df2["M8"] = df2["M8"].astype(str)
df2["M9"] = df2["M9"].astype(str)
df2["ProductCD"] = df2["ProductCD"].astype(str)

# Initializing Encoder
number = preprocessing.LabelEncoder()

# Encoding
df2["card4"] = number.fit_transform(df2["card4"])
df2["card6"] = number.fit_transform(df2["card6"])
df2["P_emaildomain"] = number.fit_transform(df2["P_emaildomain"])
df2["R_emaildomain"] = number.fit_transform(df2["R_emaildomain"])
df2["M1"] = number.fit_transform(df2["M1"])
df2["M2"] = number.fit_transform(df2["M2"])
df2["M3"] = number.fit_transform(df2["M3"])
df2["M4"] = number.fit_transform(df2["M4"])
df2["M5"] = number.fit_transform(df2["M5"])
df2["M6"] = number.fit_transform(df2["M6"])
df2["M7"] = number.fit_transform(df2["M7"])
df2["M8"] = number.fit_transform(df2["M8"])
df2["M9"] = number.fit_transform(df2["M9"])
df2["ProductCD"] = number.fit_transform(df2["ProductCD"])

df2.head()
```

Out[13]:

	TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card1	card2	card3	c
0	2987000	0	86400	68.5	4	13926	-999.0	150.0	
1	2987001	0	86401	29.0	4	2755	404.0	150.0	
2	2987002	0	86469	59.0	4	4663	490.0	150.0	
3	2987003	0	86499	50.0	4	18132	567.0	150.0	
4	2987004	0	86506	50.0	1	4497	514.0	150.0	

5 rows × 394 columns

In [14]:

```
df2.shape
```

Out[14]:

```
(590540, 394)
```

In [15]:

```
# Using iloc for merging the datasets later on
df2 = df2.iloc[:100000]
df2.shape
```

Out[15]:

```
(100000, 394)
```

In [16]:

```
# Merging two dataframes to one big dataframe
df = pd.merge(df1, df2, on = "TransactionID", how = "outer")

df.head()
```

Out[16]:

	TransactionID	id_01	id_02	id_03	id_04	id_05	id_06	id_07	id_08	id_09	...	V
0	2987004	0.0	70787.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	...	
1	2987008	-5.0	98945.0	-999.0	-999.0	0.0	-5.0	-999.0	-999.0	-999.0	...	
2	2987010	-5.0	191631.0	0.0	0.0	0.0	0.0	-999.0	-999.0	0.0	...	-9
3	2987011	-5.0	221832.0	-999.0	-999.0	0.0	-6.0	-999.0	-999.0	-999.0	...	-9
4	2987016	0.0	7460.0	0.0	0.0	1.0	0.0	-999.0	-999.0	0.0	...	

5 rows × 434 columns

In [17]:

```
# To analyze and drop Rows/Columns with Null values
df = df.dropna()
```

In [18]:

```
df.shape
```

Out[18]:

```
(41446, 434)
```

In [20]:

```
# Defining X and y and Splitting data
from sklearn.model_selection import train_test_split

# Columns used as predictors
X = df.drop(["isFraud"], axis = 1).values

y = df["isFraud"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0, test_siz
e = 0.25)
X_train
```

Out[20]:

```
array([[ 3.026214e+06, -5.000000e+00,  6.295030e+05, ..., -9.990000e+02,
        -9.990000e+02, -9.990000e+02],
       [ 3.036770e+06, -5.000000e+00,  7.141100e+04, ...,  0.000000e+00,
         0.000000e+00,  0.000000e+00],
       [ 3.063830e+06, -5.000000e+00,  9.270900e+04, ...,  0.000000e+00,
         0.000000e+00,  0.000000e+00],
       ...,
       [ 3.066731e+06, -6.000000e+01,  2.551200e+05, ...,  0.000000e+00,
         0.000000e+00,  0.000000e+00],
       [ 3.048269e+06,  0.000000e+00,  8.778000e+04, ...,  0.000000e+00,
         0.000000e+00,  0.000000e+00],
       [ 2.999115e+06, -1.500000e+01,  1.775040e+05, ..., -9.990000e+02,
        -9.990000e+02, -9.990000e+02]])
```

In [22]:

```
# Applying Random Forest
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(criterion = "entropy")
model.fit(X_train, y_train)

model.score(X_test, y_test)
```

Out[22]:

```
0.9830148619957537
```

In [ ]:

```

```