7. Applications of Data Mining for Fraud Detection - Part 2: Online Retailer

In this video, we will walk through a comprehensive process of applying machine learning techniques using real-life data. We will train test and evaluate using an Unseprvised learning method.

Import necessary libraries

In [6]:
```
import pandas as pd
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
```

Import the dataset

In [10]:
```
# Load the data
df = pd.read_excel('Online Retail.xlsx')
```

In [11]:
```
df
```

Out[11]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

541909 rows × 8 columns

Data Preprocessing

In [3]:
```
# Convert the InvoiceDate to datetime and extract the total amount spent on each transaction
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['TotalAmount'] = df['UnitPrice'] * df['Quantity']

# We will only use 'TotalAmount' for our analysis
df = df[['TotalAmount']]

# Handle missing values
df = df.dropna()
```

Scale the features

In [4]:
```
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

Model Training

In [7]:
```
# We will use Isolation Forest for anomaly detection
clf = IsolationForest(contamination=0.01)
clf.fit(df_scaled)
```

Out[7]:
```
IsolationForest(contamination=0.01)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Predict the anomalies in the data

In [8]:
```
df['anomaly'] = clf.predict(df_scaled)

# Filter and sort the anomalies
anomalies = df.loc[df['anomaly'] == -1]
anomalies = anomalies.sort_values('TotalAmount', ascending=False)

# Print the anomalies
print(anomalies)
```

```
        TotalAmount  anomaly
540421    168469.60       -1
61619      77183.60       -1
222680     38970.00       -1
15017      13541.33       -1
299982     11062.06       -1
...             ...      ...
43702     -16888.02       -1
524602    -17836.46       -1
222681    -38970.00       -1
61624     -77183.60       -1
540422   -168469.60       -1

[5361 rows x 2 columns]
```

In [ ]:
```

```