import all module

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score,f1_score, confusion_matrix,accuracy_score
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer, make_column_transformer
```

In [2]:
```
#read data
df=pd.read_excel("/users/macofa/pythonassignment5/CC-3Y-Transactions.xlsx")
#change index
df.index=pd.to_datetime(df["Date"],unit='ms')
del df["Date"]
```

In [3]:
```
print(df.head())
```

```
                            City  Card Type Exp Type Gender  Amount
Date
2014-10-29           Delhi, India       Gold    Bills      F   82475
2014-08-22  Greater Mumbai, India   Platinum    Bills      F   32555
2014-08-27       Bengaluru, India     Silver    Bills      F  101738
2014-04-12  Greater Mumbai, India  Signature    Bills      F  123424
2015-05-05       Bengaluru, India       Gold    Bills      F  171574
```

In [4]:
```
print("Info", df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 26052 entries, 2014-10-29 to 2013-10-19
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   City       26052 non-null  object
 1   Card Type  26052 non-null  object
 2   Exp Type   26052 non-null  object
 3   Gender     26052 non-null  object
 4   Amount     26052 non-null  int64 
dtypes: int64(1), object(4)
memory usage: 1.2+ MB
Info None
```

In [5]:
```
#df.dropna(subset="Amount",inplace=True)
print("Number of Nodata (NaN) ", df.isna().sum())
```

```
Number of Nodata (NaN)  City         0
Card Type    0
Exp Type     0
Gender       0
Amount       0
dtype: int64
```

In [6]:
```
df["DayNumber"]=df.index.dayofweek
```

In [7]:
```
#First create a column  named Fraud
df["Fraud"]=0
# the number of Day of Week is 6 , that day is Sunday , it is not  possible to do Transaction
df["Fraud"][df.index.dayofweek==6]=1
```

```
<ipython-input-7-fe8856b68d90>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df["Fraud"][df.index.dayofweek==6]=1
```

In [8]:
```
#Check for imbalance in data
Fraud= df[df['Fraud'] == 1]
NoFraud = df[df['Fraud'] == 0]
print(" No Fraud  Detection Cases")
print("----------------------")
print(NoFraud.Amount.describe(),"\n")
#True Detection Cases
print("Fraud  Detection Cases")
print("----------------------")
print(Fraud.Amount.describe(),"\n")
```

```
 No Fraud  Detection Cases
----------------------
count     22240.000000
mean     156406.834802
std      102731.259585
min        1005.000000
25%       77111.000000
50%      153555.500000
75%      228044.750000
max      998077.000000
Name: Amount, dtype: float64

Fraud  Detection Cases
----------------------
count      3812.000000
mean     156438.973505
std      104992.933718
min        1026.000000
25%       77275.500000
50%      150321.500000
75%      228091.750000
max      996754.000000
Name: Amount, dtype: float64
```

In [9]:
```
# Select all columns except the last for all rows
feature_cols=['City', 'Card Type', 'Exp Type', 'Gender', 'Amount', "DayNumber"]
X = df[feature_cols]
# Select the last column of all rows
Y = df['Fraud']

print(X.values.shape)
print(Y.values.shape)
```

```
(26052, 6)
(26052,)
```

In [10]:
```
#train_test_split method 80% for calibration  20 % for validation
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [11]:
```
#convert category to codes
preprocess = make_column_transformer((OneHotEncoder(handle_unknown='ignore'),['City', 'Card Type', 'Exp Type', 'Gender' , "DayNumber"]),
                                     (StandardScaler(), ["Amount"]))
```

In [12]:
```
#model Classifier
classifier = make_pipeline(
    preprocess,
    RandomForestClassifier( random_state=0))
classifier.fit(X_train,Y_train)
```

Out[12]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(transformers=[('onehotencoder',
                                                  OneHotEncoder(handle_unknown='ignore'),
                                                  ['City', 'Card Type',
                                                   'Exp Type', 'Gender',
                                                   'DayNumber']),
                                                 ('standardscaler',
                                                  StandardScaler(),
                                                  ['Amount'])])),
                ('randomforestclassifier',
                 RandomForestClassifier(random_state=0))])
```

In [13]:
```
predicted=classifier.predict(X_test)
print("\npredicted values :\n",predicted)
```

```
predicted values :
 [1 0 0 ... 0 0 1]
```

In [14]:
```
print(" confusion matrix ")
cf=confusion_matrix(Y_test, predicted)
print(cf)
```

```
 confusion matrix
[[4469    0]
 [   0  742]]
```

In [15]:
```
# #Overall Accuracy
OA = accuracy_score(Y_test, predicted) * 100
print("\nThe accuracy score  : ",OA)
#
```

```
The accuracy score  :  100.0
```

In [16]:
```
# #Precision
print('precision')
# Precision = TP / (TP + FP) (Where TP = True Positive, TN = True Negative, FP = F
precision = precision_score(Y_test, predicted, pos_label=1)
print(precision)
```

```
precision
1.0
```

In [17]:
```
#Recall
print('recall')
# Recall = TP / (TP + FN)
recall = recall_score(Y_test, predicted, pos_label=1)
print(recall)
```

```
recall
1.0
```