Chapter 5 - Ex2: Loan Prediction

Cho dữ liệu trong thư mục loan_prediction-1 chứa thông tin giao dịch cho vay của một ngân hàng

- Trong phạm vi bài này, chúng ta chỉ xem xét tập tin X_train.csv và Y_train.csv, dùng để huấn luyện mô hình dự đoán cho vay hay không cho vay. Phân tích thông tin sơ bộ về dữ liệu X_train (với các dữ liệu numeric như: ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History). Trực quan hóa dữ liệu.
- Để việc dự đoán tốt hơn cần phải kiểm tra và chuẩn hóa dữ liệu. Hãy chọn một phương pháp để chuẩn hóa dữ liệu dựa trên thông tin nêu trên. Trực quan hóa kết quả so sánh trước và sau chuẩn hóa.

Gợi ý:

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# Đọc dữ liệu. Tìm hiểu thông tin sơ bộ về dữ liệu
train_data = pd.read_csv("loan_prediction-1/X_train.csv")
train_data.head()
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
0	LP001032	Male	No	0	Graduate	No	4950	
1	LP001824	Male	Yes	1	Graduate	No	2882	
2	LP002928	Male	Yes	0	Graduate	No	3000	
3	LP001814	Male	Yes	2	Graduate	No	9703	
4	LP002244	Male	Yes	0	Graduate	No	2333	

In [3]:

```
train_data_sub = train_data[['ApplicantIncome', 'CoapplicantIncome',
                             'LoanAmount', 'Loan_Amount_Term',
                             'Credit_History']]
train_data_sub.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 384 entries, 0 to 383
Data columns (total 5 columns):
ApplicantIncome      384 non-null int64
CoapplicantIncome    384 non-null float64
LoanAmount           384 non-null int64
Loan_Amount_Term     384 non-null int64
Credit_History       384 non-null int64
dtypes: float64(1), int64(4)
memory usage: 15.1 KB
```

In [4]:

```
# Kiểm tra dữ liệu null
print(train_data_sub.isnull().sum())
# => Không có dữ liệu null
```

```
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
dtype: int64
```

In [6]:

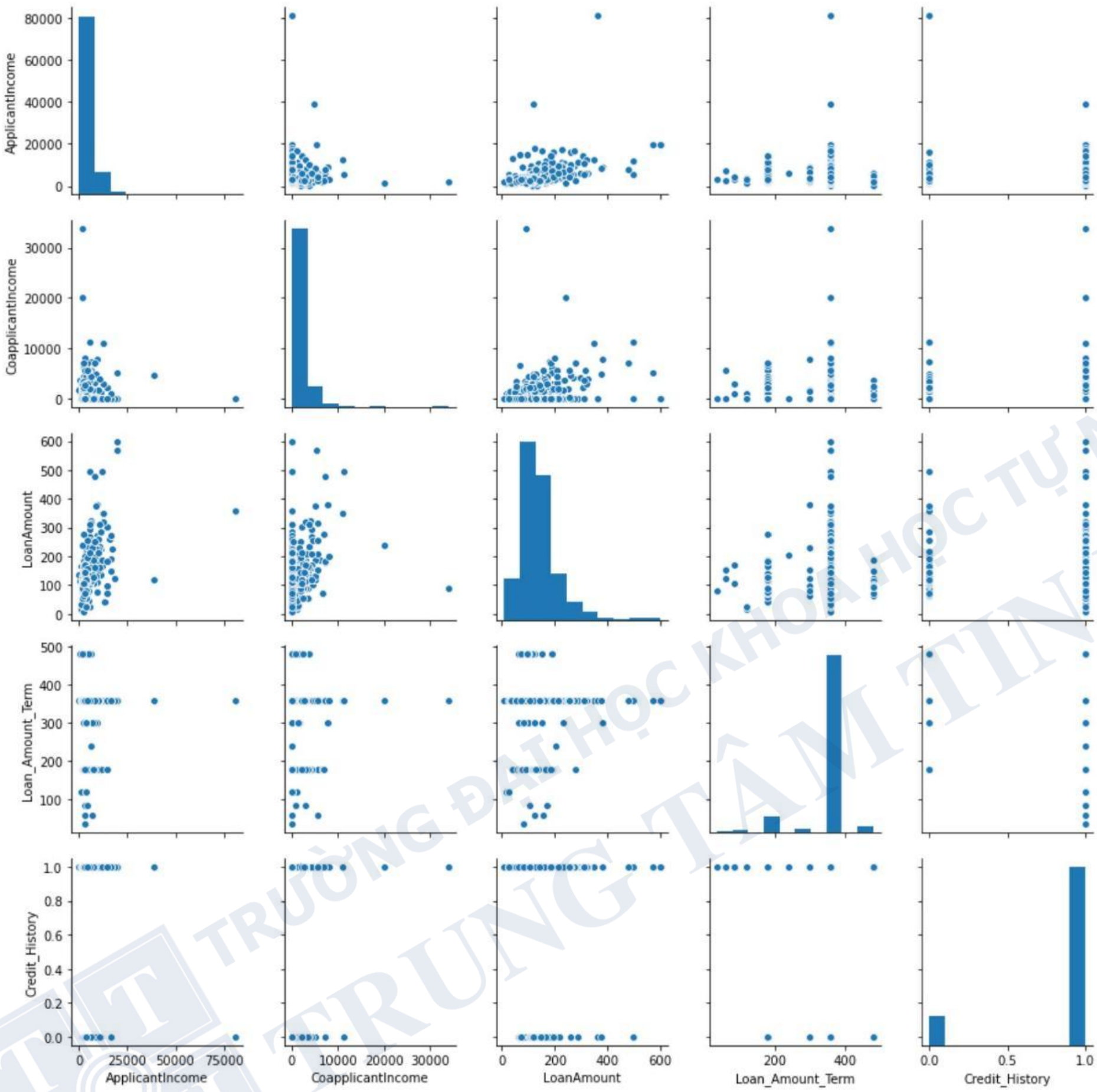
```
# Credit_History là dữ liệu nhị phân nên không cần chuẩn hóa
# Loan_Amount_Term: đơn vị là tháng
# 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount' với don vị là USD
```

In [5]:

```
sns.pairplot(train_data_sub)
```

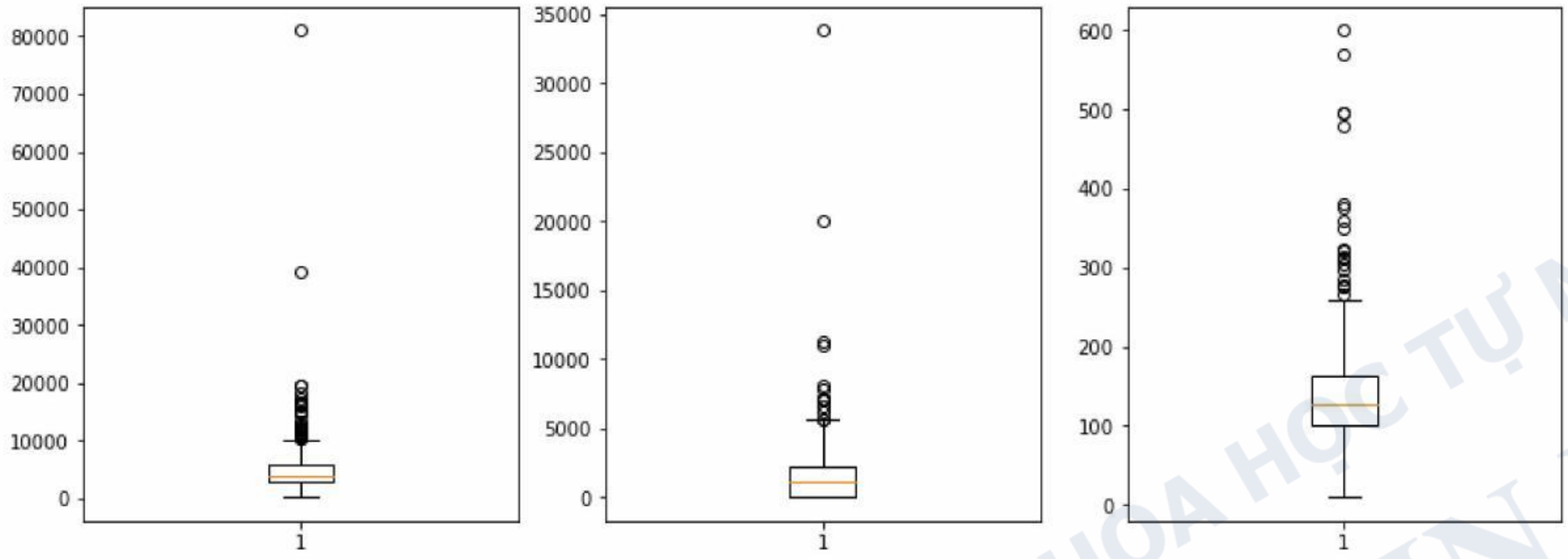
Out[5]:

```
<seaborn.axisgrid.PairGrid at 0x1de9a705d68>
```



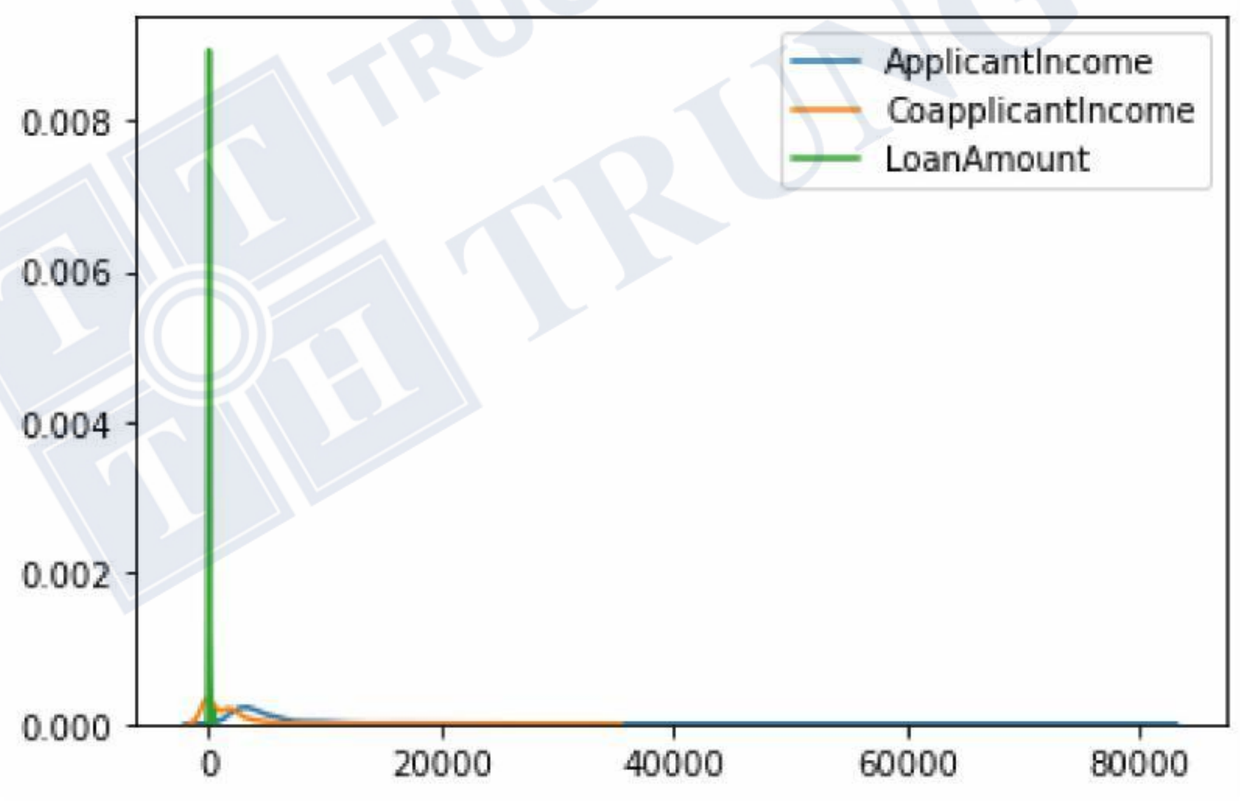
In [6]:

```
# Phân tích đơn biến: trực quan hóa, kiểm tra dữ liệu outlier
# Trực quan hóa dữ liệu cho từng biến liên tục
plt.figure(figsize=(14,5))
plt.subplot(1,3,1)
plt.boxplot(train_data_sub.ApplicantIncome)
plt.subplot(1,3,2)
plt.boxplot(train_data_sub.CoapplicantIncome)
plt.subplot(1,3,3)
plt.boxplot(train_data_sub.LoanAmount)
plt.show()
# => các biến đều có outlier trên
```



In [7]:

```
sns.kdeplot(train_data_sub.ApplicantIncome)
sns.kdeplot(train_data_sub.CoapplicantIncome)
sns.kdeplot(train_data_sub.LoanAmount)
plt.show()
```



- Có một thang dữ liệu khác nhau như sau: ApplicantIncome: 0 -> 80000USD, LoanAmount: 0->600USD

Giải pháp 1:

- Dữ liệu không phân phối chuẩn => Xem xét việc loại các outliers => Sau đó áp dụng MinMaxScaler

MinMaxScaler

- Noted: Học viên tự loại bỏ outlier trước khi áp dụng MinMaxScaler

In [12]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [13]:

```
min_max=MinMaxScaler()
```

In [14]:

```
 X_train_minmax=min_max.fit_transform(train_data_sub)
```

In [15]:

```
X_train_minmax[:4]
```

Out[15]:

```
array([[0.0593692 , 0.        , 0.1962775 , 0.72972973, 1.        ],
       [0.03379097, 0.054467  , 0.1928934 , 1.        , 1.        ],
       [0.03525046, 0.10095458, 0.07952623, 0.32432432, 1.        ],
       [0.11815708, 0.        , 0.17428088, 0.72972973, 1.        ]])
```

In [16]:

```
df_train_minmax = pd.DataFrame(X_train_minmax,
                               columns=['ApplicantIncome', 'CoapplicantIncome',
                                        'LoanAmount', 'Loan_Amount_Term',
                                        'Credit_History'])
df_train_minmax.head()
```

Out[16]:

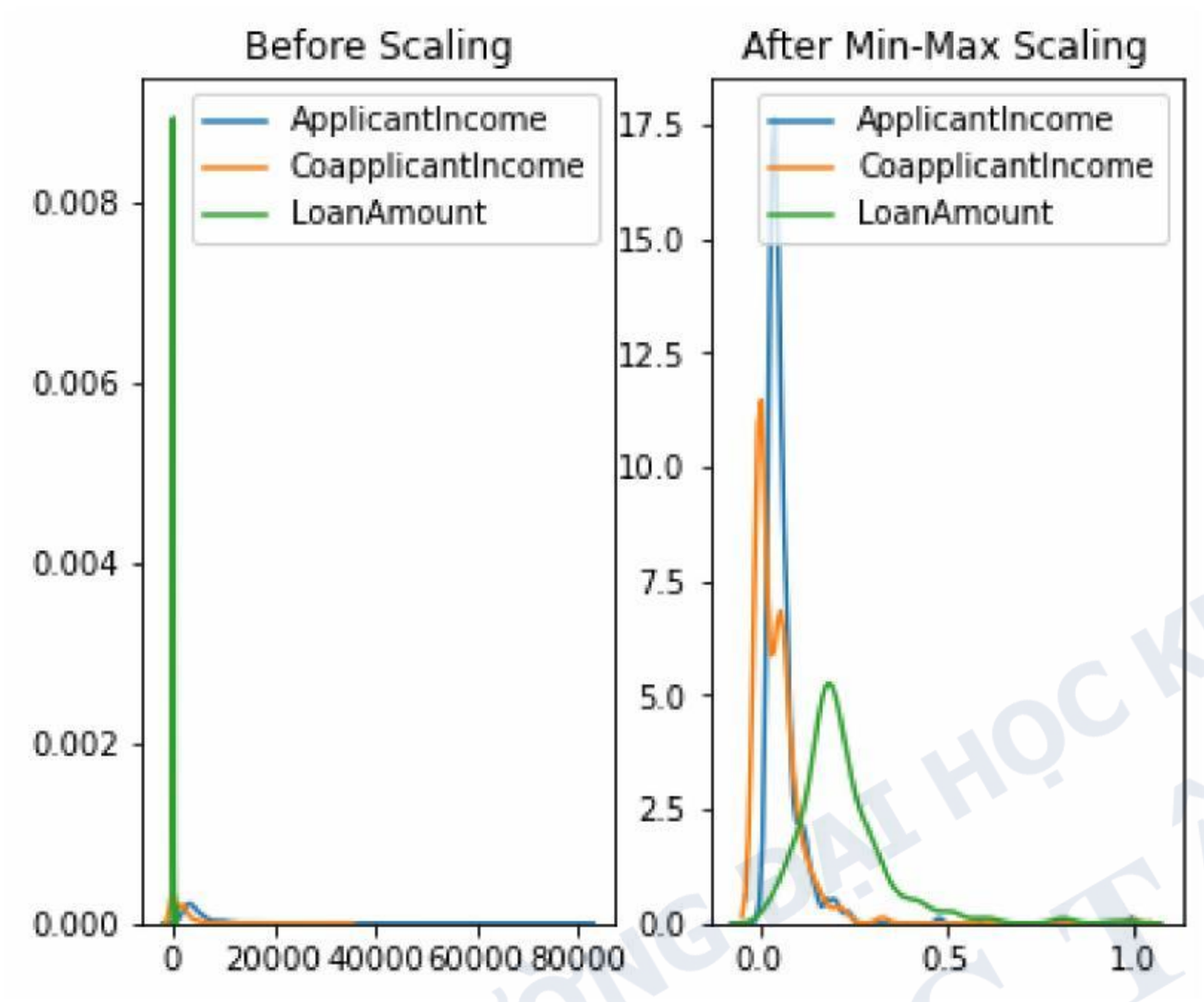
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.059369	0.000000	0.196277	0.729730	1.0
1	0.033791	0.054467	0.192893	1.000000	1.0
2	0.035250	0.100955	0.079526	0.324324	1.0
3	0.118157	0.000000	0.174281	0.729730	1.0
4	0.027001	0.071431	0.214890	0.729730	1.0

In [17]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5))
ax1.set_title('Before Scaling')
sns.kdeplot(train_data_sub.ApplicantIncome, ax=ax1)
sns.kdeplot(train_data_sub.CoapplicantIncome, ax=ax1)
sns.kdeplot(train_data_sub.LoanAmount, ax=ax1)

ax2.set_title('After Min-Max Scaling')
sns.kdeplot(df_train_minmax.ApplicantIncome, ax=ax2)
sns.kdeplot(df_train_minmax.CoapplicantIncome, ax=ax2)
sns.kdeplot(df_train_minmax.LoanAmount, ax=ax2)

plt.show()
```



Giải pháp 2:

- Dùng RobustScaler nếu muốn giữ lại outliers

RobustScaler

In [18]:

```
from sklearn.preprocessing import RobustScaler
```

In [19]:

```
rbs=RobustScaler()
```

In [20]:

```
 X_train_rbs=rbs.fit_transform(train_data_sub)
```

In [21]:

```
X_train_rbs[:4]
```

Out[21]:

```
array([[ 3.61722160e-01, -5.26059533e-01, -4.66926070e-02,
         0.00000000e+00,  0.00000000e+00],
       [-3.46315159e-01,  2.89697909e-01, -7.78210117e-02,
         1.20000000e+02,  0.00000000e+00],
       [-3.05914577e-01,  9.85946664e-01, -1.12062257e+00,
        -1.80000000e+02,  0.00000000e+00],
       [ 1.98904391e+00, -5.26059533e-01, -2.49027237e-01,
         0.00000000e+00,  0.00000000e+00]])
```

In [22]:

```
df_train_rbs = pd.DataFrame(X_train_rbs,
                            columns=['ApplicantIncome', 'CoapplicantIncome',
                                     'LoanAmount', 'Loan_Amount_Term',
                                     'Credit_History'])
df_train_rbs.head()
```

Out[22]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.361722	-0.526060	-0.046693	0.0	0.0
1	-0.346315	0.289698	-0.077821	120.0	0.0
2	-0.305915	0.985947	-1.120623	-180.0	0.0
3	1.989044	-0.526060	-0.249027	0.0	0.0
4	-0.534281	0.543765	0.124514	0.0	0.0

In [23]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(6, 5))
ax1.set_title('Before Scaling')
sns.kdeplot(train_data_sub.ApplicantIncome, ax=ax1)
sns.kdeplot(train_data_sub.CoapplicantIncome, ax=ax1)
sns.kdeplot(train_data_sub.LoanAmount, ax=ax1)

ax2.set_title('After Robust Scaling')
sns.kdeplot(df_train_rbs.ApplicantIncome, ax=ax2)
sns.kdeplot(df_train_rbs.CoapplicantIncome, ax=ax2)
sns.kdeplot(df_train_rbs.LoanAmount, ax=ax2)

plt.show()
```