IT4930: Nhập môn Khoa học dữ liệu

BÁO CÁO BÀI THỰC HÀNH TIỀN XỬ LÝ DỮ LIỆU

Họ tên sinh viên: Trần Gia Định

MSSV: 20235036

MỤC TIÊU BÀI LÀM

Hoàn thành tiền xử lý dữ liệu cho tập dữ liệu về tình trạng khoản vay của các sinh viên

BƯỚC 1: IMPORT CÁC THƯ VIỆN CẦN THIẾT

In [15]:
```
import numpy as np
import pandas as pd
from sklearn import preprocessing
```

BƯỚC 2: GIỚI THIỆU VỀ TẬP DỮ LIỆU

VẤN ĐỀ THỰC TẾ: Dự đoán tình trạng khoản vay

Mô tả bài toán

Dựa trên thông tin từ sinh viên như giới tính, trình độ học vấn, số tiền vay để dự đoán tình trạng vay của sinh viên

Mục tiêu

Xây dựng mô hình Học máy để dự đoán tình trạng vay vốn của sinh viên

BƯỚC 3: TẢI VÀ ĐỌC DỮ LIỆU

BƯỚC 3.1. Tải dữ liệu

In [16]:
```
data = pd.read_csv('train_u6lujuX_CVtuZ9i.csv')
# In ra 5 dòng đầu tiên của dữ liệu
data.head()
```

Out[16]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

BƯỚC 3.2. Kiểm tra các kích thước của dữ liệu

In [17]:
```
print("Thông tin chung của dữ liệu:")
print('-' * 50)
print(f"Số dòng dữ liệu: {data.shape[0]}")
print(f"Số thuộc tính: {data.shape[1]}")
print(f"Số ô dữ liệu: {data.shape[0] * data.shape[1]}")
print(f"Số chiều dữ liệu: {data.shape}")
```

```
Thông tin chung của dữ liệu:
--------------------------------------------------
Số dòng dữ liệu: 614
Số thuộc tính: 13
Số ô dữ liệu: 7982
Số chiều dữ liệu: (614, 13)
```

BƯỚC 4: KHAI PHÁ DỮ LIỆU

BƯỚC 4.1. Tổng quan về dữ liệu

In [18]:
```
# Tổng quan dữ liệu
print("THÔNG TIN DỮ LIỆU CHI TIẾT:")
print("=" * 50)

# Đưa ra thông tin về DataFrame
data.info()

# Giải thích kết quả
```

```
print("\nGIẢI THÍCH:")
print("• Non-Null Count: Số giá trị không bị thiếu")
print("• Dtype: Kiểu dữ liệu (object = text, int64 = integer, float64 = float)")
print("• Memory usage: Bộ nhớ dùng bởi DataFrame")
```

```
THÔNG TIN DỮ LIỆU CHI TIẾT:
==================================================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB

GIẢI THÍCH:
• Non-Null Count: Số giá trị không bị thiếu
• Dtype: Kiểu dữ liệu (object = text, int64 = integer, float64 = float)
• Memory usage: Bộ nhớ dùng bởi DataFrame
```

BƯỚC 4.2. Thống kê chi tiết cho dữ liệu số

In [19]:
```
# Thống kê chi tiết của các cột dữ liệu số
print("THỐNG KÊ CHI TIẾT CỦA CÁC CỘT DỮ LIỆU SỐ")
print("=" * 50)

# Tính toán thống kê mô tả
numeric_stats = data.describe()
print(numeric_stats)

# Quan sát
print("\nQuan sát một số đặc trưng kiểu số:")
print("• ApplicantIncome: phạm vi từ 150 đến 81000, trung bình ~5400, phương sai ca
print("• CoapplicantIncome: phạm vi từ 0 đến ~42000, trung bình ~1600")
```

```
THỐNG KÊ CHI TIẾT CỦA CÁC CỘT DỮ LIỆU SỐ
==================================================
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       614.000000         614.000000  592.000000         600.00000
mean       5403.459283        1621.245798  146.412162         342.00000
std        6109.041673        2926.248369   85.587325          65.12041
min         150.000000           0.000000    9.000000          12.00000
25%        2877.500000           0.000000  100.000000         360.00000
50%        3812.500000        1188.500000  128.000000         360.00000
75%        5795.000000        2297.250000  168.000000         360.00000
max       81000.000000       41667.000000  700.000000         480.00000

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

Quan sát một số đặc trưng kiểu số:
- ApplicantIncome: phạm vi từ 150 đến 81000, trung bình ~5400, phương sai cao ~6100
- CoapplicantIncome: phạm vi từ 0 đến ~42000, trung bình ~1600

BƯỚC 4.3. Khám phá dữ liệu phân loại

In [21]:
```python
# Khám phá dữ liệu phân loại
print("PHÂN TÍCH DỮ LIỆU PHÂN LOẠI")
print("=" * 50)

categorical_columns = data.select_dtypes(include='object').columns

for col in categorical_columns:
    print(f"\nCột '{col}':")

    # Các giá trị phân biệt
    unique_values = data[col].unique()
    # Đưa ra 5 giá trị phân biệt đầu tiên
    print(f"  Giá trị phân biệt: {unique_values[:5]}")
    print(f"  Số giá trị phân biệt: {len(unique_values)}")

    # Số lượng giá trị và phân phối của nó
    print("  PHÂN PHỐI:")
    value_counts = data[col].value_counts()
    # In ra 5 giá trị phân biệt đầu tiên và phân phối của nó
    i = 0
    for value, count in value_counts.items():
        i += 1
        percentage = (count / len(data)) * 100
        print(f"    • {value}: {count} ({percentage:.1f}%)")
        if i == 5: break

print("\nCÁC QUAN SÁT QUAN TRỌNG:")
```

```
print("• Tất cả dữ liệu phân loại cần được chuyển về dạng số")
print("• Đầu ra dự đoán: Y (yes) hoặc N (no)")
```

```
PHÂN TÍCH DỮ LIỆU PHÂN LOẠI
==================================================

Cột 'Loan_ID':
  Giá trị phân biệt: ['LP001002' 'LP001003' 'LP001005' 'LP001006' 'LP001008']
  Số giá trị phân biệt: 614
  PHÂN PHỐI:
    • LP001002: 1 (0.2%)
    • LP001003: 1 (0.2%)
    • LP001005: 1 (0.2%)
    • LP001006: 1 (0.2%)
    • LP001008: 1 (0.2%)

Cột 'Gender':
  Giá trị phân biệt: ['Male' 'Female' nan]
  Số giá trị phân biệt: 3
  PHÂN PHỐI:
    • Male: 489 (79.6%)
    • Female: 112 (18.2%)

Cột 'Married':
  Giá trị phân biệt: ['No' 'Yes' nan]
  Số giá trị phân biệt: 3
  PHÂN PHỐI:
    • Yes: 398 (64.8%)
    • No: 213 (34.7%)

Cột 'Dependents':
  Giá trị phân biệt: ['0' '1' '2' '3+' nan]
  Số giá trị phân biệt: 5
  PHÂN PHỐI:
    • 0: 345 (56.2%)
    • 1: 102 (16.6%)
    • 2: 101 (16.4%)
    • 3+: 51 (8.3%)

Cột 'Education':
  Giá trị phân biệt: ['Graduate' 'Not Graduate']
  Số giá trị phân biệt: 2
  PHÂN PHỐI:
    • Graduate: 480 (78.2%)
    • Not Graduate: 134 (21.8%)

Cột 'Self_Employed':
  Giá trị phân biệt: ['No' 'Yes' nan]
  Số giá trị phân biệt: 3
  PHÂN PHỐI:
    • No: 500 (81.4%)
    • Yes: 82 (13.4%)

Cột 'Property_Area':
  Giá trị phân biệt: ['Urban' 'Rural' 'Semiurban']
  Số giá trị phân biệt: 3
  PHÂN PHỐI:
    • Semiurban: 233 (37.9%)
    • Urban: 202 (32.9%)
```

```
    • Rural: 179 (29.2%)

Cột 'Loan_Status':
  Giá trị phân biệt: ['Y' 'N']
  Số giá trị phân biệt: 2
  PHÂN PHỐI:
    • Y: 422 (68.7%)
    • N: 192 (31.3%)

CÁC QUAN SÁT QUAN TRỌNG:
• Tất cả dữ liệu phân loại cần được chuyển về dạng số
• Đầu ra dự đoán: Y (yes) hoặc N (no)
```

BƯỚC 5: TIỀN XỬ LÝ DỮ LIỆU

MỤC TIÊU TỔNG QUAN

Chuyển đổi dữ liệu thô thành dữ liệu sẵn sàng cho các thuật toán ML

1. Phân tách dữ liệu thành X (đặc trưng) và y (mục tiêu)
2. Mã hóa dữ liệu phân loại
3. Chuẩn hóa dữ liệu số

Tại sao chúng ta cần xử lý dữ liệu trước khi áp dụng mô hình ML/DL?

Thuật toán Học máy chỉ hiểu dữ liệu số. Tuy nhiên, dữ liệu thô bao gồm:

- **Dữ liệu văn bản**: "Male", "Female", "Graduate", "Not Graduate"
- **Thang điểm hỗn hợp**: ApplicantIncome (150-81000), CoapplicantIncome (0-42000)
- **Định dạng khác nhau**: Cần thống nhất định dạng

Sau khi xử lý trước:

- **Tất cả dữ liệu số**: Dữ liệu ML đã sẵn sàng
- **Cùng thang điểm**: Không còn vấn đề về phạm vi giá trị
- **Sẵn sàng ML**: Dữ liệu có thể được đưa trực tiếp vào mô hình ML

Làm sạch dữ liệu

In [22]:
```
# ------------------------------
# 1) Khám phá phần trăm giá trị NaN trong tất cả các cột
# ------------------------------
num_rows = data.shape[0]
print("Phần trăm giá trị NaN:")

for col in data.columns:
    percentage = data[col].isna().sum() / num_rows * 100
    print(f"Cột {col}:\n {percentage:.2f}%")
```

```
Phần trăm giá trị NaN:
Cột Loan_ID:
 0.00%
Cột Gender:
 2.12%
Cột Married:
 0.49%
Cột Dependents:
 2.44%
Cột Education:
 0.00%
Cột Self_Employed:
 5.21%
Cột ApplicantIncome:
 0.00%
Cột CoapplicantIncome:
 0.00%
Cột LoanAmount:
 3.58%
Cột Loan_Amount_Term:
 2.28%
Cột Credit_History:
 8.14%
Cột Property_Area:
 0.00%
Cột Loan_Status:
 0.00%
```

Vì số lượng giá trị bị thiếu khá nhỏ so với số lượng hàng, chúng ta sẽ điền vào các giá trị bị thiếu này theo mean (với dữ liệu số) và mode (với dữ liệu phân loại)

In [23]:
```python
# ----------------------------
# 2) Xử lý các giá trị bị thiếu
# ----------------------------
# Điền các giá trị dạng số với mean
num_cols = data.select_dtypes(include=[np.number]).columns            # Chọn các cộ
for col in num_cols:
    data[col] = data[col].fillna(data[col].mean())

# Điền các giá trị phân loại với mode
cat_cols = data.select_dtypes(exclude=[np.number]).columns            # Chọn các cộ
for col in cat_cols:
    mode_val = data[col].mode().iloc[0]                               # Lấy giá trị
    data[col] = data[col].fillna(mode_val)

print("=== SAU KHI XỬ LÝ DỮ LIỆU THIẾU ===")
data.info()  # Hiển thị thông tin về các đặc trưng của dữ liệu sau khi điền các giá
```

```
=== SAU KHI XỬ LÝ DỮ LIỆU THIẾU ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             614 non-null    object
 2   Married            614 non-null    object
 3   Dependents         614 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      614 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    float64
 9   Loan_Amount_Term   614 non-null    float64
 10  Credit_History     614 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [24]:

```python
# ------------------------------
# 3) Loại bỏ các hàng có giá trị trùng lặp
# ------------------------------
before = data.shape[0]               # Số hàng trước khi loại bỏ trùng lặp
data = data.drop_duplicates()        # Loại bỏ các hàng trùng lặp
after = data.shape[0]                # Số hàng sau khi loại bỏ trùng lặp
print(f"Đã loại bỏ {before - after} hàng trùng lặp.\n")

# ------------------------------
# Dữ liệu sau khi làm sạch
# ------------------------------
print("Như ta có thể thấy, dữ liệu không có hàng nào trùng lặp")
print("=== DỮ LIỆU SAU KHI LÀM SẠCH ===")
print(data)
```

```
Đã loại bỏ 0 hàng trùng lặp.

Như ta có thể thấy, dữ liệu không có hàng nào trùng lặp
=== DỮ LIỆU SAU KHI LÀM SẠCH ===
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0  146.412162             360.0
1               4583             1508.0  128.000000             360.0
2               3000                0.0   66.000000             360.0
3               2583             2358.0  120.000000             360.0
4               6000                0.0  141.000000             360.0
..               ...                ...         ...               ...
609             2900                0.0   71.000000             360.0
610             4106                0.0   40.000000             180.0
611             8072              240.0  253.000000             360.0
612             7583                0.0  187.000000             360.0
613             4583                0.0  133.000000             360.0

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]
```

BƯỚC 5.1. Chia dữ liệu thành 2 tập: X và y

X: Các đặc trưng sử dụng cho việc dự đoán

y: Giá trị mục tiêu cần được dự đoán

Trong phần này, chúng ta sẽ sử dụng phương thức '.values', phương thức này chuyển đổi một DataFrame thành một mảng NumPy, điều này thích hợp cho bài toán của chúng ta do mảng NumPy xử lý nhanh hơn DataFrame

In [25]:
```
# Tạo ra ma trận thuộc tính X
print("=== CHIA DỮ LIỆU THÀNH TẬP X VÀ Y ===")
print("=" * 50)

# Chọn các cột và chuyển đổi thành mảng NumPy
# Bỏ cột 'Loan_ID' vì nó không mang lại khả năng dự đoán cho các thuật toán ML
X = data.drop(columns=['Loan_Status', 'Loan_ID']).values

print("=== Thông tin về ma trận đặc trưng ===")
print(f"   Chiều dữ liệu: {X.shape}")
print(f"   Kiểu dữ liệu: {X.dtype}")
print(f"   5 hàng đầu tiên:")
# In ra 5 hàng đầu tiên
for i in range(5):
    print(f"   Student {i}: {X[i]}")

print("\nGIẢI THÍCH:")
print("   • Mỗi hàng tương ứng với thông tin của 1 sinh viên")
print("   • Mỗi cột đặc tả 1 đặc trưng")
print("   • Cột 0, 1, 3, 4, 10 bao gồm văn bản → cần được mã hóa")

# In ra dữ liệu sau khi chuyển đổi sang mảng NumPy
X[0:5]
```

```
=== CHIA DỮ LIỆU THÀNH TẬP X VÀ Y ===
==================================================
=== Thông tin về ma trận đặc trưng ===
   Chiều dữ liệu: (614, 11)
   Kiểu dữ liệu: object
   5 hàng đầu tiên:
   Student 0: ['Male' 'No' '0' 'Graduate' 'No' 5849 0.0 146.41216216216216 360.0 1.0
 'Urban']
   Student 1: ['Male' 'Yes' '1' 'Graduate' 'No' 4583 1508.0 128.0 360.0 1.0 'Rural']
   Student 2: ['Male' 'Yes' '0' 'Graduate' 'Yes' 3000 0.0 66.0 360.0 1.0 'Urban']
   Student 3: ['Male' 'Yes' '0' 'Not Graduate' 'No' 2583 2358.0 120.0 360.0 1.0 'Urb
an']
   Student 4: ['Male' 'No' '0' 'Graduate' 'No' 6000 0.0 141.0 360.0 1.0 'Urban']

GIẢI THÍCH:
   • Mỗi hàng tương ứng với thông tin của 1 sinh viên
   • Mỗi cột đặc tả 1 đặc trưng
   • Cột 0, 1, 3, 4, 10 bao gồm văn bản → cần được mã hóa
```

Out[25]:
```
array([['Male', 'No', '0', 'Graduate', 'No', 5849, 0.0,
        146.41216216216216, 360.0, 1.0, 'Urban'],
       ['Male', 'Yes', '1', 'Graduate', 'No', 4583, 1508.0, 128.0, 360.0,
        1.0, 'Rural'],
       ['Male', 'Yes', '0', 'Graduate', 'Yes', 3000, 0.0, 66.0, 360.0,
        1.0, 'Urban'],
       ['Male', 'Yes', '0', 'Not Graduate', 'No', 2583, 2358.0, 120.0,
        360.0, 1.0, 'Urban'],
       ['Male', 'No', '0', 'Graduate', 'No', 6000, 0.0, 141.0, 360.0,
        1.0, 'Urban']], dtype=object)
```

BƯỚC 5.2. Label Encoding

Vì các thuật toán ML không thể hiểu dữ liệu phân loại, chúng ta cần một cách để mã hóa mỗi giá trị văn bản thành một số nguyên duy nhất.

Trong phần này, chúng ta sẽ sử dụng LabelEncoder từ thư viện sklearn để hỗ trợ thực hiện nhiệm vụ này.

In [26]:
```
print("=== LABEL ENCODING CHO DỮ LIỆU PHÂN LOẠI ===")
print("=" * 60)

X_df = data.drop(columns=['Loan_Status', 'Loan_ID'])
df = X_df.select_dtypes(include='object')

for col in df.columns:
    col_idx = X_df.columns.get_loc(col)
    # Khởi LabelEncoder
    scaler = preprocessing.LabelEncoder()
    # Fit - học các giá trị khác nhau
    le_scale = scaler.fit(X[:, col_idx])
    # Transform - chuyển dữ liệu sang dạng số
    X[:, col_idx] = le_scale.transform(X[:, col_idx])

# ===== KẾT QUẢ CUỐI CÙNG =====
print(f"\n=== KẾT QUẢ SAU MÃ HÓA ===")
print(f"   Số chiều của X: {X.shape}")
print(f"   In ra 5 dòng đầu tiên:")
for i in range(5):
    print(f"   Sinh viên {i}: {X[i]}")

print(f"\nHOÀN THÀNH: Tất cả dữ liệu đã được chuyển về dạng số!")

# Đưa ra kết quả cuối cùng
X[0:5]
```

```
=== LABEL ENCODING CHO DỮ LIỆU PHÂN LOẠI ===
============================================================

=== KẾT QUẢ SAU MÃ HÓA ===
   Số chiều của X: (614, 11)
   In ra 5 dòng đầu tiên:
   Sinh viên 0: [1 0 0 0 0 5849 0.0 146.41216216216216 360.0 1.0 2]
   Sinh viên 1: [1 1 1 0 0 4583 1508.0 128.0 360.0 1.0 0]
   Sinh viên 2: [1 1 0 0 1 3000 0.0 66.0 360.0 1.0 2]
   Sinh viên 3: [1 1 0 1 0 2583 2358.0 120.0 360.0 1.0 2]
   Sinh viên 4: [1 0 0 0 0 6000 0.0 141.0 360.0 1.0 2]

HOÀN THÀNH: Tất cả dữ liệu đã được chuyển về dạng số!
```

Out[26]:
```
array([[1, 0, 0, 0, 0, 5849, 0.0, 146.41216216216216, 360.0, 1.0, 2],
       [1, 1, 1, 0, 0, 4583, 1508.0, 128.0, 360.0, 1.0, 0],
       [1, 1, 0, 0, 1, 3000, 0.0, 66.0, 360.0, 1.0, 2],
       [1, 1, 0, 1, 0, 2583, 2358.0, 120.0, 360.0, 1.0, 2],
       [1, 0, 0, 0, 0, 6000, 0.0, 141.0, 360.0, 1.0, 2]], dtype=object)
```

BƯỚC 5.3. Chuẩn hóa dữ liệu số

Vấn đề với các thang đo khác nhau:

Thuật toán ML có thể làm sai lệch các đặc trưng với giá trị lớn hơn

Giải pháp: Chuẩn hóa Z - Score

Chuẩn hóa chuyển đổi dữ liệu thành phân phối chuẩn:

- **Trung bình (μ) = 0**: Trung bình = 0
- **Độ lệch chuẩn (σ) = 1**: Độ lệch chuẩn = 1

Lợi ích của chuẩn hóa:

1. Cân bằng: Tất cả các đặc trưng đều có tầm quan trọng như nhau
2. Tốc độ: Thuật toán hội tụ nhanh hơn
3. Độ chính xác: Kết quả dự đoán tốt hơn

In [27]:
```
# Chuẩn hóa một số đặc trưng dạng số: 'ApplicantIncome', 'CoapplicantIncome', 'Loan
print("=== CHUẨN HÓA DỮ LIỆU SỐ ===")
print("=" * 60)

# Đưa ra dữ liệu trước khi chuẩn hóa
# Vì chúng ta chuẩn hóa nhiều tính năng, để ngắn gọn, ta chỉ hiển thị thông tin 'Ap
print("TRƯỚC CHUẨN HÓA:")
print("Một số thông tin của đặc trưng 'ApplicantIncome'")
ApplicantIncome_original = X[:, 5].copy()            # Lưu bản gốc để so sánh
print(f"   5 giá trị đầu: {ApplicantIncome_original[0:5]}")
print(f"   Giá trị min: {ApplicantIncome_original.min():.2f}")
print(f"   Giá trị max: {ApplicantIncome_original.max():.2f}")
print(f"   Trung bình: {ApplicantIncome_original.mean():.2f}")
print(f"   Độ lệch chuẩn: {ApplicantIncome_original.std():.2f}")
```

```
scale_features = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amoun
for col in scale_features:
    # Bước 1: Tính mean (trung bình) và std (độ lệch chuẩn)
    col_idx = X_df.columns.get_loc(col)
    mean = X[:, col_idx].mean()
    std = X[:, col_idx].std()

    print(f"\nTÍNH TOÁN THAM SỐ:")
    print(f"   μ (mean) = {mean:.4f}")
    print(f"   σ (std)  = {std:.4f}")

    # Bước 2: Áp dụng công thức z = (x - μ)/σ
    print(f"\nÁP DỤNG CHUẨN HÓA:")
    print(f"   Công thức: z = (x - {mean:.2f}) / {std:.2f}")

    X[:, col_idx] = (X[:, col_idx] - mean) / std

# Đưa ra dữ liệu sau chuẩn hóa
# Vì chúng ta chuẩn hóa nhiều tính năng, để ngắn gọn, ta chỉ hiển thị thông tin 'Ap
print(f"\nSAU CHUẨN HÓA:")
print("Một số thông tin của đặc trưng 'ApplicantIncome'")
print(f"   5 giá trị đầu: {X[0:5, 5]}")
print(f"   Giá trị min: {X[:, 5].min():.2f}")
print(f"   Giá trị max: {X[:, 5].max():.2f}")
print(f"   Trung bình mới: {X[:, 5].mean():.6f} ≈ 0")
print(f"   Độ lệch chuẩn mới: {X[:, 5].std():.6f} ≈ 1")

# So sánh trước và sau chuẩn hóa
print(f"\nTRƯỚC VÀ SAU CHUẨN HÓA:")
print("Chỉ số | Trước | Sau")
print("-" * 45)
for i in range(5):
    print(f"     {i}       |       {ApplicantIncome_original[i]:8.2f}       |    {X

# Đưa ra ma trận đặc trưng X cuối cùng
print(f"\nMA TRẬN ĐẶC TRƯNG X SAU KHI CHUẨN HÓA:")
X[0:5]
```

```
=== CHUẨN HÓA DỮ LIỆU SỐ ===
============================================================
TRƯỚC CHUẨN HÓA:
Một số thông tin của đặc trưng 'ApplicantIncome'
   5 giá trị đầu: [5849 4583 3000 2583 6000]
   Giá trị min: 150.00
   Giá trị max: 81000.00
   Trung bình: 5403.46
   Độ lệch chuẩn: 6104.06

TÍNH TOÁN THAM SỐ:
   μ (mean) = 5403.4593
   σ (std)  = 6104.0649

ÁP DỤNG CHUẨN HÓA:
   Công thức: z = (x - 5403.46) / 6104.06

TÍNH TOÁN THAM SỐ:
   μ (mean) = 1621.2458
   σ (std)  = 2923.8645

ÁP DỤNG CHUẨN HÓA:
   Công thức: z = (x - 1621.25) / 2923.86

TÍNH TOÁN THAM SỐ:
   μ (mean) = 146.4122
   σ (std)  = 83.9690

ÁP DỤNG CHUẨN HÓA:
   Công thức: z = (x - 146.41) / 83.97

TÍNH TOÁN THAM SỐ:
   μ (mean) = 342.0000
   σ (std)  = 64.3200

ÁP DỤNG CHUẨN HÓA:
   Công thức: z = (x - 342.00) / 64.32

SAU CHUẨN HÓA:
Một số thông tin của đặc trưng 'ApplicantIncome'
   5 giá trị đầu: [0.0729908228506884 -0.1344119537834513 -0.3937473372051284
 -0.4620624697931507 0.09772843680942071]
   Giá trị min: -0.86
   Giá trị max: 12.38
   Trung bình mới: -0.000000 ≈ 0
   Độ lệch chuẩn mới: 1.000000 ≈ 1

TRƯỚC VÀ SAU CHUẨN HÓA:
Chỉ số | Trước | Sau
-------------------------------------------
     0       |        5849.00       |        0.07
     1       |        4583.00       |       -0.13
     2       |        3000.00       |       -0.39
     3       |        2583.00       |       -0.46
     4       |        6000.00       |        0.10
```

```
MA TRẬN ĐẶC TRƯNG X SAU KHI CHUẨN HÓA:
```

Out[27]:
```
array([[1, 0, 0, 0, 0, 0.0729908228506884, -0.5544873301529847,
        3.3847857674402117e-16, 0.27985054320228187, 1.0, 2],
       [1, 1, 1, 0, 0, -0.1344119537834513, -0.03873154846446161,
        -0.21927331495275829, 0.27985054320228187, 1.0, 0],
       [1, 1, 0, 0, 1, -0.3937473372051284, -0.5544873301529847,
        -0.9576409986248733, 0.27985054320228187, 1.0, 2],
       [1, 1, 0, 1, 0, -0.4620624697931507, 0.25197960169153616,
        -0.31454656445883766, 0.27985054320228187, 1.0, 2],
       [1, 0, 0, 0, 0, 0.09772843680942071, -0.5544873301529847,
        -0.06445428450537935, 0.27985054320228187, 1.0, 2]], dtype=object)
```

BƯỚC 5.4. Xử lý biến mục tiêu

Tạo vectơ y từ cột 'Loan_Status'

In [28]:
```
# Tạo vector mục tiêu y
print("=== TẠO VECTOR MỤC TIÊU ===")
print("=" * 40)

# Lấy cột 'Loan_Status' làm mục tiêu
y = data['Loan_Status']
print("THÔNG TIN MỤC TIÊU:")
print(f"   Số chiều: {y.shape}")
print(f"   Kiểu dữ liệu: {y.dtype}")
print(f"   5 giá trị đầu tiên: {list(y[0:5])}")

print(f"\nPHÂN PHỐI LỚP:")
class_counts = y.value_counts()
for drug, count in class_counts.items():
    percentage = (count/len(y)) * 100
    print(f"   {drug}: {count} ({percentage:.1f}%)")

print(f"\nNHẬN XÉT:")
print(f"   • Có {len(class_counts)} trạng thái vay khác nhau")
print(f"   • Đây là bài toán phân loại 2 lớp")

# Đưa ra một số giá trị ban đầu
y[0:5]
```

```
=== TẠO VECTOR MỤC TIÊU ===
========================================
THÔNG TIN MỤC TIÊU:
   Số chiều: (614,)
   Kiểu dữ liệu: object
   5 giá trị đầu tiên: ['Y', 'N', 'Y', 'Y', 'Y']

PHÂN PHỐI LỚP:
   Y: 422 (68.7%)
   N: 192 (31.3%)

NHẬN XÉT:
   • Có 2 trạng thái vay khác nhau
   • Đây là bài toán phân loại 2 lớp
```

Out[28]:
```
0    Y
1    N
2    Y
3    Y
4    Y
Name: Loan_Status, dtype: object
```

Mã hóa biến mục tiêu

Hầu hết các thuật toán ML yêu cầu biến mục tiêu là số, không phải văn bản. Do đó, chúng ta cần mã hóa biến mục tiêu từ văn bản sang số nguyên.

In [29]:
```
# Mã hóa biến mục tiêu y
print("=== MÃ HÓA BIẾN MỤC TIÊU Y ===")
print("=" * 50)

# Fit các trạng thái vay
le_status = scaler.fit(y.unique())

print("THÔNG TIN MÃ HÓA NHÃN:")
print(f"   Các lớp học được: {le_status.classes_}")

# Hiển thị mapping
print(f"\nBẢNG MAPPING:")
for i, status in enumerate(le_status.classes_):
    print(f"   {status} → {i}")

# Biểu diễn encoding
print(f"\nTRƯỚC ENCODING:")
print(f"   5 giá trị đầu tiên: {list(y[0:5])}")

y_encoded = le_status.transform(y)

print(f"\n  SAU ENCODING:")
print(f"   5 giá trị đầu tiên: {y_encoded[0:5]}")
print(f"   Kiểu dữ liệu: {y_encoded.dtype}")
print(f"   Số chiều: {y_encoded.shape}")

# Cập nhật y
y = y_encoded

print(f"\nHOÀN THÀNH: Biến mục tiêu y đã được chuyển về dạng số!")

# Kiểm tra phân phối của y sau encoding
print(f"\nPHÂN PHỐI BIẾN MỤC TIÊU SAU MÃ HÓA:")
unique, counts = np.unique(y, return_counts=True)
for val, count in zip(unique, counts):
    status_name = le_status.classes_[val]
    percentage = (count/len(y)) * 100
    print(f"   {val} ({status_name}): {count} ({percentage:.1f}%)")
```

```
=== MÃ HÓA BIẾN MỤC TIÊU Y ===
==================================================
THÔNG TIN MÃ HÓA NHÃN:
   Các lớp học được: ['N' 'Y']

BẢNG MAPPING:
   N → 0
   Y → 1

TRƯỚC ENCODING:
   5 giá trị đầu tiên: ['Y', 'N', 'Y', 'Y', 'Y']

  SAU ENCODING:
   5 giá trị đầu tiên: [1 0 1 1 1]
   Kiểu dữ liệu: int64
   Số chiều: (614,)

HOÀN THÀNH: Biến mục tiêu y đã được chuyển về dạng số!

PHÂN PHỐI BIẾN MỤC TIÊU SAU MÃ HÓA:
   0 (N): 192 (31.3%)
   1 (Y): 422 (68.7%)
```

In [30]:

```python
# Đưa ra kết quả cuối cùng
print("=== KẾT QUẢ CUỐI CÙNG SAU TIỀN XỬ LÝ DỮ LIỆU ===")
print("=" * 50)

print("Ma trận đặc trưng X:")
print(f"   Số chiều: {X.shape}")
print(f"   5 dòng đầu tiên:")
for i in range(5):
    print(f"     Sinh viên {i}: {X[i]}")

print(f"\nVector y (Mục tiêu):")
print(f"   Số chiều: {y.shape}")
print(f"   5 giá trị đầu tiên: {y[0:5]}")

print(f"\nDỮ LIỆU ĐÃ SẴN SÀNG CHO ML!")
print(f"    Tất cả các giá trị đều là dạng số")
print(f"    Dữ liệu đã được chuẩn hóa")
print(f"    Không có dữ liệu nào bị thiếu")
print(f"    Định dạng đã sẵn sàng cho các thuật toán ML!")

# In ra các giá trị đầu tiên của y
y[0:5]
```

```
=== KẾT QUẢ CUỐI CÙNG SAU TIỀN XỬ LÝ DỮ LIỆU ===
==================================================
Ma trận đặc trưng X:
   Số chiều: (614, 11)
   5 dòng đầu tiên:
     Sinh viên 0: [1 0 0 0 0 0.0729908228506884 -0.5544873301529847 3.38478576744021
17e-16
 0.27985054320228187 1.0 2]
     Sinh viên 1: [1 1 1 0 0 -0.1344119537834513 -0.03873154846446161 -0.21927331495
275829
 0.27985054320228187 1.0 0]
     Sinh viên 2: [1 1 0 0 1 -0.3937473372051284 -0.5544873301529847 -0.957640998624
8733
 0.27985054320228187 1.0 2]
     Sinh viên 3: [1 1 0 1 0 -0.4620624697931507 0.25197960169153616 -0.314546564458
83766
 0.27985054320228187 1.0 2]
     Sinh viên 4: [1 0 0 0 0 0.09772843680942071 -0.5544873301529847 -0.064454284505
37935
 0.27985054320228187 1.0 2]

Vector y (Mục tiêu):
   Số chiều: (614,)
   5 giá trị đầu tiên: [1 0 1 1 1]

DỮ LIỆU ĐÃ SẴN SÀNG CHO ML!
    Tất cả các giá trị đều là dạng số
    Dữ liệu đã được chuẩn hóa
    Không có dữ liệu nào bị thiếu
    Định dạng đã sẵn sàng cho các thuật toán ML!
```

Out[30]: array([1, 0, 1, 1, 1])

KẾT LUẬN

Đầu ra dữ liệu sau khi tiền xử lý là ma trận NumPy có giá trị ở các cột đều là dạng số, các cột (đặc trưng) đã được chuẩn hóa và các đặc trưng phân loại đã được mã hóa về dạng số. Ngoài ra, dữ liệu bị thiếu cũng đã được điền thông qua mean (đối với dữ liệu số) và mode (đối với dữ liệu phân loại). Các hàng trùng lặp cũng đã được xử lý.

Đầu ra này là hoàn toàn phù hợp với các thuật toán ML, do dữ liệu cho ML là dữ liệu sạch, đầy đủ, toàn số, và được chuẩn hoá để mô hình học hiệu quả.