

# BỘ GIÁO DỤC VÀ ĐÀO TẠO

# TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

## KHOA CÔNG NGHỆ THÔNG TIN



# BÁO CÁO CUỐI KỲ MÔN PHÂN TÍCH DỮ LIỆU

## ĐỀ TÀI:

# PHÂN TÍCH NHỮNG YẾU TỐ ẢNH HƯỞNG ĐẾN TIỀM NĂNG KHÁCH HÀNG CHO VAY CÁ NHÂN

**GVHD:** Ths. Nguyễn Văn Thành

Lớp: Thứ 5 (tiết 7 – 10)

Mã lớp: DAAN436277 23 2 01

Sinh viên thực hiện: (nhóm 5)

Nguyễn Đức Duy	MSSV: 21133020
Nguyễn Trọng Dũng	MSSV: 21133021
Đỗ Ngọc Hân	MSSV: 21133030
Huỳnh Gia Hân	MSSV: 21133031
Trần Thị Ngọc Trang	MSSV: 21133109

Thành phố Hồ Chí Minh, tháng 05, năm 2024

## NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

### Họ và tên sinh viên thực hiện

- Nguyễn Đức Duy - 21133020
- Nguyễn Trọng Dũng - 21133021
- Đỗ Ngọc Hân - 21133030
- Huỳnh Gia Hân - 21133031
- Trần Thị Ngọc Trang - 21133109

**Chuyên ngành:** Kỹ thuật dữ liệu (Data Engineering)

**Đề tài:** Phân tích những yếu tố ảnh hưởng đến tiềm năng khách hàng cho vay cá nhân.

**Môn học:** Phân tích dữ liệu (Data Analysis)

### Nhận xét

.....  
.....  
.....  
.....  
.....

Tp HCM, / / 2024

Giảng viên hướng dẫn

(Tên và chữ ký)

Nguyễn Văn Thành

## PHÂN CÔNG NHIỆM VỤ NHÓM 5

Nhiệm vụ	Đức Duy	Trọng Dũng	Ngọc Hân	Gia Hân	Ngọc Trang
Tìm kiếm tập dữ liệu			x		
Giới thiệu về tập dữ liệu	x				
Phân tích mối tương quan		x			
Kiểm tra Describe các biến có giá trị là số				x	
<b>Tiền xử lý dữ liệu</b>			<b>x</b>	<b>x</b>	<b>x</b>
Xử lý nhiễu			x		
Xử lý dữ liệu ngoại lai (Outlier)				x	
Xử lý giá trị bị thiếu (NULL)					x
Xử lý các giá trị trùng lặp (Duplicated)					x
Biến đổi đặc trưng					x
<b>Phân tích dữ liệu thăm dò (EDA) và Trực quan hóa</b>	<b>x</b>		<b>x</b>		<b>x</b>
Phân tích chung	x				
Tìm hiểu cột Personal Loan	x				
Phân tích thông tin cá nhân khách hàng: Age, Experience, và Income			x		
Phân tích thông tin cá nhân khách hàng: Family, Education, Securities Account					x
Phân tích thông tin cá nhân khách hàng: CD Account, Online, CreditCard					x
Phân tích thông tin cá nhân khách hàng: CCAvg, Mortgage					x

Nhiệm vụ	Đức Duy	Trọng Dũng	Ngọc Hân	Gia Hân	Ngọc Trang
<b>Thiết lập các mô hình phân loại cho dự đoán “Person Loan”</b>		x		x	
Kiểm tra phân phối của giá trị cần phân loại		x			
Ma trận tương quan giữa các cột		x			
Phân chia dữ liệu thành 2 tập Train và Test				x	
<b>Lựa chọn mô hình phân tích</b>	x	x	x	x	x
Decision Tree Model		x			
XGBClassifier Model	x				
KneighborsClassifier Model			x		
LogisticRegression Model				x	
RandomForest Model					x
<b>Đánh giá và kết luận</b>		x	x	x	
Đánh giá các mô hình			x		
Đánh giá các mô hình trên cả hai trường hợp		x			
Kết hợp với phân tích đưa ra kết luận				x	

## LỜI CẢM ƠN

Bài báo cáo này là sản phẩm của một quá trình học tập và làm việc nhóm. Để có thể hoàn thành bài báo cáo, chúng em đã nhận được rất nhiều sự hỗ trợ từ thầy và các bạn. Do đó nhóm chúng em xin được phép gửi lời cảm ơn chân thành và sâu sắc nhất đến:

- 1) *Thầy Nguyễn Văn Thành – giảng viên bộ môn Phân tích dữ liệu (Data Analysis), khoa Công nghệ Thông tin trường Đại học Sư phạm Kỹ thuật thành phố Hồ Chí Minh. Nhóm xin cảm ơn sự quan tâm và giúp đỡ tận tình của thầy trong suốt quá trình giảng dạy. Cảm ơn thầy đã luôn giải đáp những thắc mắc cũng như đưa ra những nhận xét, góp ý giúp nhóm thực hiện cải thiện chất lượng công việc của nhóm.*
- 2) *Thư viện trường Đại học Sư phạm Kỹ thuật thành phố Hồ Chí Minh, nơi cung cấp môi trường học tập, nghiên cứu và làm việc nhóm để chúng em có thể hoàn thành tốt báo cáo của nhóm mình.*
- 3) *Các bạn học cùng lớp đã có những nhận xét, đóng góp về mặt kiến thức lần tinh thần cho nhóm.*

Đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian ngắn, với những hạn chế khác về mặt kiến thức, kỹ thuật và kinh nghiệm trong việc thực hiện báo cáo. Do đó, trong quá trình làm đề tài có những thiếu sót là điều không thể tránh khỏi. Chúng em rất mong nhận được những ý kiến đóng góp quý báu của thầy để kiến thức của chúng em được hoàn thiện hơn và làm tốt hơn nữa trong những lần sau. Chúng em xin chân thành cảm ơn!

# MỤC LỤC

<b>PHẦN 1: MỞ ĐẦU.....</b>	<b>1</b>
<b>1. Lý do chọn đề tài .....</b>	<b>1</b>
<b>2. Đối tượng và phạm vi nghiên cứu chọn.....</b>	<b>1</b>
<b>PHẦN 2: NỘI DUNG .....</b>	<b>3</b>
<b>CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ ĐỀ TÀI.....</b>	<b>3</b>
<b>1.1. Giới thiệu về tập dữ liệu.....</b>	<b>3</b>
<b>1.2. Mô tả chi tiết về tập dữ liệu .....</b>	<b>3</b>
<b>1.3. Giới thiệu các công cụ được sử dụng.....</b>	<b>4</b>
<b>CHƯƠNG 2: PHÂN TÍCH NHỮNG YÊU TỐ ẢNH HƯỞNG ĐẾN TIỀM NĂNG KHÁCH HÀNG CHO VAY CÁ NHÂN .....</b>	<b>5</b>
<b>2.1. Các thư viện được sử dụng trong bài phân tích .....</b>	<b>5</b>
<b>2.2. Thông tin về tập dữ liệu .....</b>	<b>7</b>
2.2.1. Đọc và hiển thị 5 dòng đầu tiên của tập dữ liệu .....	7
2.2.2. Kiểm tra số dòng và số cột của tập dữ liệu .....	7
2.2.3. Chi tiết các cột trong tập dữ liệu .....	8
<b>2.3. Phân tích mối tương quan.....</b>	<b>10</b>
<b>2.4. Kiểm tra Describe các biến có giá trị là số .....</b>	<b>12</b>
<b>2.5. Tiền xử lý dữ liệu.....</b>	<b>13</b>
2.5.1. Xử lý nhiễu.....	13
2.5.2. Xử lý dữ liệu ngoại lai (Outlier) .....	15
2.4.3. Xử lý dữ liệu bị thiếu (NULL).....	17
2.5.4. Xử lý các giá trị trùng lặp (Duplicated) .....	17
2.5.5. Biến đổi đặc trưng .....	18

<b>2.6. Phân tích dữ liệu thăm dò (EDA) và Trực quan hóa.....</b>	<b>19</b>
2.6.1. Phân tích chung .....	19
2.6.2. Tìm hiểu cột Personal Loan.....	21
2.6.3. Tìm hiểu các thông tin liên quan/ảnh hưởng đến cột Personal Loan .	21
<b>2.7. Thiết lập các mô hình phân loại cho dự đoán “Personal Loan” .....</b>	<b>30</b>
2.7.1. Kiểm tra phân phối của giá trị cần phân loại.....	30
2.7.2. Ma trận tương quan giữa các cột.....	31
2.7.3. Phân chia dữ liệu thành 2 tập Train và Test .....	32
2.7.4. Decission Tree Model.....	33
2.7.5. XGBClassifier Model.....	38
2.7.6. KNeighborsClassifier Model .....	42
2.7.7. LogisticRegression Model .....	46
2.7.8. RandomForest Model .....	50
<b>CHƯƠNG 3: ĐÁNH GIÁ MÔ HÌNH VÀ ĐƯA RA KẾT LUẬN SAU KHI PHÂN TÍCH.....</b>	<b>55</b>
<b>3.1. Đánh giá mô hình .....</b>	<b>55</b>
<b>3.2. Kết luận sau khi phân tích .....</b>	<b>59</b>
<b>3.3. Xây dựng một số chương trình cho các đối tượng có xu hướng quan tâm đến vay cá nhân.....</b>	<b>60</b>
<b>PHẦN 3: KẾT LUẬN.....</b>	<b>63</b>
<b>1. KẾT QUẢ ĐẠT ĐƯỢC .....</b>	<b>63</b>
<b>1.1. Về kiến thức .....</b>	<b>63</b>
<b>1.2. Về việc thực hiện dự án .....</b>	<b>63</b>
<b>2. ƯU ĐIỂM, HẠN CHÉ CỦA ĐỀ TÀI.....</b>	<b>63</b>

<b>2.1. Ưu điểm.....</b>	<b>63</b>
<b>2.2. Hạn chế .....</b>	<b>63</b>
<b>3. HƯỚNG PHÁT TRIỂN .....</b>	<b>63</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>65</b>

# **PHẦN 1: MỞ ĐẦU**

## **1. Lý do chọn đề tài**

Trong kỷ nguyên phát triển của xã hội ngày nay, hầu như bất cứ nơi nào cũng cần xuất hiện các giao dịch về tiền tệ. Có thể nói, nhu cầu đời sống của con người không thể thiếu về vấn đề vật chất, cụ thể hơn là tiền bạc. Đối với các gia đình, cá nhân có thu nhập vừa phải, họ có thể cung cấp được niềm vui vật chất cho bản thân mình. Tuy nhiên, để có thể được ngân hàng tin tưởng cho vay cá nhân, khách hàng đó phải có nhiều yếu tố tin tưởng, đủ để ngân hàng tin rằng họ có thể là người mà ngân hàng có thể cho vay. Nhận thấy điều đó, nhóm chúng em quyết định chọn “Phân tích những yếu tố ảnh hưởng đến tiềm năng khách hàng cho vay cá nhân” làm đề tài cuối kỳ môn Phân tích dữ liệu. Với mong muốn hỗ trợ ngân hàng tìm ra được những khách hàng có thể cho vay cá nhân.

## **2. Đối tượng và phạm vi nghiên cứu chọn**

Đối tượng nghiên cứu của đồ án là dữ liệu về những yếu tố thông tin của khách hàng để xem xét ngân hàng cho vay cá nhân.

Phạm vi nghiên cứu:

- Tìm hiểu về tập dữ liệu về những yếu tố thông tin của khách hàng để xem xét cho vay cá nhân.
- Thực hiện phân tích dữ liệu trên Jupyter Notebook.
- Thực hiện các mô hình học máy để xác định tiềm năng khách hàng mà ngân hàng có thể cho vay cá nhân.

## **3. Kết quả dự kiến đạt được**

Hiểu rõ về tập dữ liệu được dùng để phân tích.

Thực hiện các phân tích cơ bản trên tập dữ liệu.

Thực hiện các mô hình học máy trên tập dữ liệu.

## **PHẦN 2: NỘI DUNG**

### **CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ ĐỀ TÀI**

#### **1.1. Giới thiệu về tập dữ liệu.**

Ngân hàng (Thera Bank) có lượng khách hàng phần lớn trong đó là khách hàng gửi tiền (depositors) với các mức gửi tiền khác nhau. Số lượng khách hàng là người đi vay (asset customers) khá ít và ngân hàng quan tâm đến việc nhanh chóng mở rộng nhóm đối tượng khách hàng này để mang lại nhiều hoạt động cho vay và kiếm tiền nhiều hơn thông qua lãi suất cho vay. Đặc biệt các nhà quản lý muốn tìm hiểu các cách chuyển đổi khách hàng nợ thành khách hàng cho vay cá nhân (trong khi vẫn giữ họ làm người gửi tiền). Ngân hàng đã triển khai một chiến dịch vào năm ngoái dành cho khách hàng nợ và đạt được tỷ lệ chuyển đổi tốt với mức thành công trên 9%. Điều này đã khuyến khích bộ phận tiếp thị bán lẻ đưa ra các chiến dịch để tiếp thị mục tiêu tốt hơn.

Bộ phận muốn xây dựng mô hình giúp họ xác định những khách hàng tiềm năng có xác suất vay cá nhân cao hơn. Điều này sẽ làm tăng tỷ lệ thành công đồng thời giảm chi phí của chiến dịch.

Link dataset: <https://www.kaggle.com/datasets/mahnazarjmand/bank-personal-loan>

#### **1.2. Mô tả chi tiết về tập dữ liệu**

Tập dữ liệu Bank\_Personal\_Loan\_Modelling.csv có 5000 dòng và 14 cột: ID, Age, Experience, Income, Zip Code, Family, Ccavg, Education, Mortgage, Personal Loan, Securities Account, CD Account, Online và CreditCard. (Chi tiết về các cột dữ liệu sẽ được tìm hiểu ở phần phân tích).

### **1.3. Giới thiệu các công cụ được sử dụng**

Visual Code và Jupyter Notebook để thực hiện các dòng lệnh code Python.

# CHƯƠNG 2: PHÂN TÍCH NHỮNG YẾU TỐ ẢNH HƯỚNG ĐẾN TIỀM NĂNG KHÁCH HÀNG CHO VAY CÁ NHÂN

## 2.1. Các thư viện được sử dụng trong bài phân tích

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Untitled (Workspace), Python 3.11.9.
- Explorer:** UNTITLED (WORKSPACE) contains NLP-clustering-word--Vietnamese-S..., BankLoanStatusAnalysis, KiemTra, Bank-Loan, FinalProject, Project, Code (containing Nhom5final.ipynb), Data (containing Bank\_Personal\_Loan\_Modelling.csv), -WRL0003.tmp, MSSV\_Report.docx, MSSV\_Report.docx.
- Code Cell:** 1. Nhập thư viện

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

from matplotlib.colors import LinearSegmentedColormap
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, roc_curve, roc_auc_score
from sklearn.metrics import classification_report
from scipy import stats
from sklearn.metrics import confusion_matrix

%matplotlib inline
# Tắt cảnh báo
warnings.filterwarnings('ignore')
```

- Code Cell:** 2. Thông tin về tập dữ liệu

1. numpy: là thư viện cung cấp hỗ trợ cho các mảng và ma trận lớn, cùng với các hàm toán học hiệu quả để thao tác trên các cấu trúc dữ liệu này.
2. pandas: cung cấp các cấu trúc dữ liệu và công cụ phân tích dữ liệu mạnh mẽ, như DataFrame và Series.
3. matplotlib: là thư viện vẽ đồ thị 2D linh hoạt và mạnh mẽ.
4. seaborn là thư viện trực quan hóa dữ liệu dựa trên Matplotlib, cung cấp giao diện mức cao để tạo các biểu đồ đẹp và nhiều thông tin.
5. warnings: được sử dụng để thao tác cho hiển thị các cảnh báo trong Python.
6. matplotlib.colors: hỗ trợ tạo các bản đồ màu tùy chỉnh.
7. Scikit-Learn:
  - model\_selection import train\_test\_split: Chia dữ liệu thành tập huấn luyện và kiểm tra

- linear\_model import LogisticRegression: mô hình Hồi quy Logistic để phân loại.
- neighbors import KneighborsClassifier: mô hình Phân loại dựa trên thuật toán k-nearest neighbors.
- tree import DecisionTreeClassifier: mô hình Xây dựng cây quyết định.
- ensemble import RandomForestClassifier: mô hình Phân loại dựa trên rừng ngẫu nhiên, một tập hợp của nhiều cây quyết định.
- metrics import accuracy\_score, roc\_curve, roc\_auc\_score, classification\_report, confusion\_matrix: Đánh giá mô hình qua các chỉ số như độ chính xác, đường cong ROC, AUC, báo cáo phân loại và ma trận nhầm lẫn.

## 8. XGBoost:

- xgboost import XGBClassifier: là một thuật toán học máy mạnh mẽ và tối ưu hóa hiệu suất dựa trên gradient boosting.

## 9. SciPy:

- scipy import stats: là thư viện bổ sung cho NumPy, cung cấp các công cụ và hàm cho khoa học máy tính và kỹ thuật.

## 2.2. Thông tin về tập dữ liệu

### 2.2.1. Đọc và hiển thị 5 dòng đầu tiên của tập dữ liệu

The screenshot shows a Jupyter Notebook interface with a red box highlighting the code and output for reading the first 5 rows of a CSV file. The code is:

```
In [2]: df = pd.read_csv('./Bank_Personal_Loan_Modelling.csv')
df.head(5)
```

The output shows a DataFrame with 5 rows and 14 columns:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1

Below this, another section is shown with a red box:

### 2.2. Kiểm tra tổng số dòng và số cột của tập dữ liệu

```
In [3]: df.shape
```

```
Out[3]: (5000, 14)
```

### 2.2.2. Kiểm tra số dòng và số cột của tập dữ liệu

The screenshot shows a Jupyter Notebook interface with a red box highlighting the code and output for checking the total number of rows and columns. The code is:

```
In [3]: df.shape
```

```
Out[3]: (5000, 14)
```

The output shows the dimensions of the DataFrame:

Bộ dữ liệu gồm:

- \* 5000 dòng
- \* 14 cột

Below this, another section is shown with a red box:

### 2.3. Chi tiết các cột trong tập dữ liệu

```
In [4]: display(df.columns)
```

```
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg', 'Education', 'Mortgage', 'Personal Loan', 'Securities Account', 'CD Account', 'Online', 'CreditCard'],  
      dtype='object')
```

Như vậy, tập dữ liệu Bank\_Personal\_Loan\_Modelling.csv có 5000 dòng dữ liệu, và 14 cột thuộc tính.

### 2.2.3. Chi tiết các cột trong tập dữ liệu

The screenshot shows a Jupyter Notebook interface. In the top navigation bar, there are tabs for 'Home Page - Select or create a notebook...', 'Nhom5 - Jupyter Notebook', and 'MSSV\_Report.docx - Google Tài liệu...'. Below the tabs, the title 'jupyter Nhom5 Last Checkpoint: 11 hours ago (autosaved)' is displayed. The main area shows code execution results:

```
Out[3]: (5000, 14)

Bộ dữ liệu gồm:
* 5000 dòng
* 14 cột
```

Below this, a section titled '2.3. Chi tiết các cột trong tập dữ liệu' contains the following code:

```
In [4]: display(df.columns)
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
       'Education', 'Mortgage', 'Personal Loan', 'Securities Account',
       'CD Account', 'Online', 'CreditCard'],
      dtype='object')
```

A red box highlights this code block. Below it, a table titled 'Thông tin về tập dữ liệu:' provides a summary of the columns:

STT	Cột	Mô tả
1	ID	Mã của khách hàng
2	Age	Tuổi của khách hàng

The table continues with more rows, but they are not fully visible in the screenshot.

Về tổng quan, 14 cột trong tập dữ liệu có ý nghĩa như sau:

STT	Cột	Mô tả
1	ID	Mã của khách hàng
2	Age	Tuổi của khách hàng
3	Experience	Số năm kinh nghiệm làm việc của khách hàng Thu nhập hàng năm của khách hàng (nghìn)
4	Income	Thu nhập hàng năm của khách hàng (nghìn)
5	Zip Code	Địa chỉ nhà theo Mã bưu điện(ZIP)
6	Family	Tổng số người trong gia đình của khách hàng
7	Ccavg	Trung bình chi tiêu bằng thẻ tín dụng hàng tháng
8	Education	Trình độ học vấn (1: Bằng cử nhân, 2: Bằng thạc sĩ, 3:Bằng cấp chuyên nghiệp)
9	Mortgage	Giá trị thuê chắp nhà(nghìn)
10	Personal Loan	Khách hàng có chấp nhận khoản vay cá nhân được cung cấp trong dihj vụ trước hay không? 0:

		Không, 1: Có
11	Securities Account	Khách hàng hiện đang có tài khoản chứng khoán tại ngân hàng không? (0: Không, 1: Có)
12	CD Account	Khách hàng có chứng nhận tài khoản khoán tiền gửi(CD) tại ngân hàng không? (0: Không, 1: Có )
13	Online	Khách hàng có sử dụng các dịch vụ ngân hàng trực tuyến không? (0: Không, 1: Có)
14	CreditCard	Khách hàng có sử dụng thẻ tín dụng do ngân hàng phát hành không? (0: Không, 1: Có)

```
In [5]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               5000 non-null    int64  
 1   Age              5000 non-null    int64  
 2   Experience       5000 non-null    int64  
 3   Income           5000 non-null    int64  
 4   ZIP Code         5000 non-null    int64  
 5   Family           5000 non-null    int64  
 6   CCAvg            5000 non-null    float64 
 7   Education        5000 non-null    int64  
 8   Mortgage          5000 non-null    int64  
 9   Personal Loan     5000 non-null    int64  
 10  Securities Account 5000 non-null    int64  
 11  CD Account        5000 non-null    int64  
 12  Online             5000 non-null    int64  
 13  CreditCard         5000 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 547.0 KB
```

• Mỗi giá trị của ID là duy nhất và thực tế nó chỉ là một mã định danh không mang thông tin có giá trị cho mô hình. Vì vậy, nhóm đã quyết định loại bỏ cột này.

Về cấu trúc của tập dữ liệu:

- Số lượng hàng và cột là 5000 dòng và 14 cột
- Tất cả các cột đều có dữ liệu kiểu số, trong đó có 13 cột là kiểu số nguyên (int64) và 1 cột là kiểu số thực (float64)
- Không có dữ liệu khuyết (thiếu) trong DataFrame vì tất cả các cột đều đủ 5000 giá trị

Ta thấy rằng cột đầu tiên, ID chỉ chứa mã định danh mỗi khách hàng, không có giá trị thông tin gì cho mục đích xây dựng mô hình => loại bỏ cột này

```

In [6]: df.drop('ID', axis=1, inplace=True)

```

Mỗi giá trị của ID là duy nhất và thực tế nó chỉ là một mã định danh không mang thông tin có giá trị cho mô hình. Vì vậy, nhóm đã quyết định loại bỏ cột này.

In [6]: `df.drop('ID', axis=1, inplace=True)`

**3. Phân tích mối tương quan**

In [7]: `# Xác định bản màu  
gray_white = LinearSegmentedColormap.from_list('gray_white', [(0, (1, 1, 1)), (1, (0.5, 0.5, 0.5))])  
gray_white_r = gray_white.reversed()`

In [\*]: `# Tính toán tương quan Spearman`

## 2.3. Phân tích mối tương quan

```

In [6]: df.drop('ID', axis=1, inplace=True)

```

Mỗi giá trị của ID là duy nhất và thực tế nó chỉ là một mã định danh không mang thông tin có giá trị cho mô hình. Vì vậy, nhóm đã quyết định loại bỏ cột này.

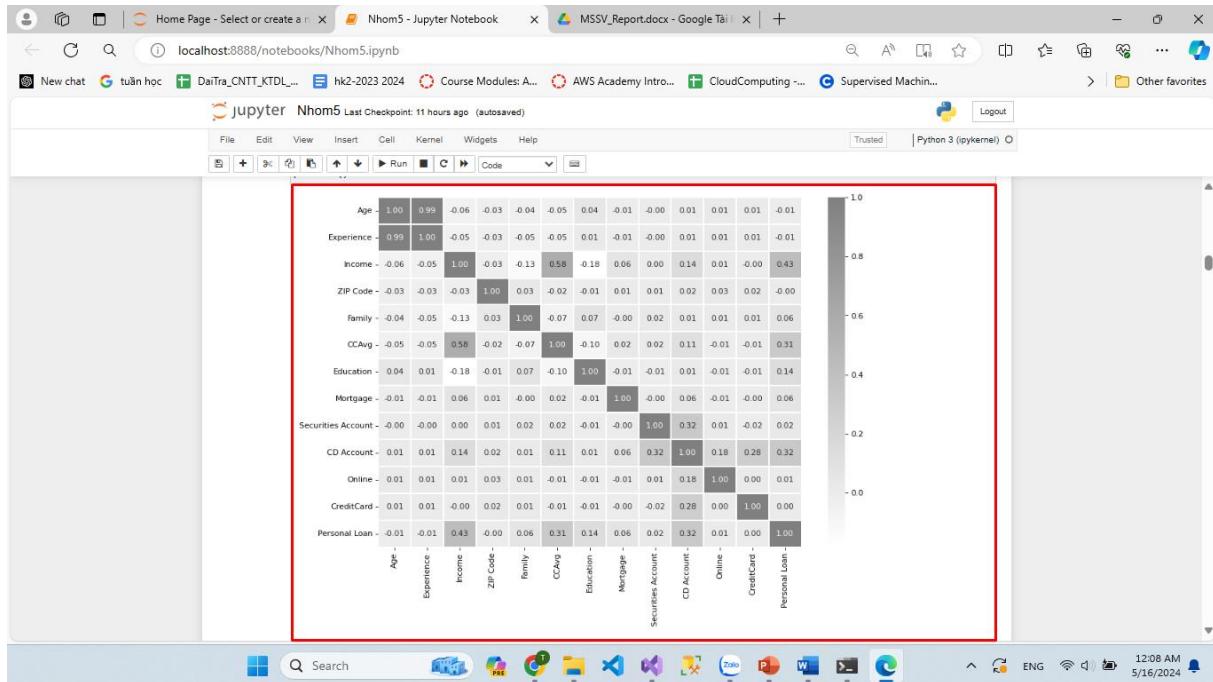
In [6]: `df.drop('ID', axis=1, inplace=True)`

**3. Phân tích mối tương quan**

In [7]: `# Xác định bản màu  
gray_white = LinearSegmentedColormap.from_list('gray_white', [(0, (1, 1, 1)), (1, (0.5, 0.5, 0.5))])  
gray_white_r = gray_white.reversed()`

In [\*]: `# Tính toán tương quan Spearman  
target = 'Personal Loan'  
df_ordered = pd.concat([df.drop(target, axis=1), df[target]], axis=1)  
corr = df_ordered.corr(method='spearman')  
  
# Vẽ heatmap  
plt.figure(figsize=(12, 8), dpi=80)  
sns.heatmap(corr, annot=True, cmap=gray_white, fmt='.2f', linewidths=0.2)  
plt.show()`

	Age	Experience	...	...	...	...	...	...	...	...	...	...	
Age	1.00	0.99	-0.06	-0.03	-0.04	-0.05	0.04	-0.01	-0.00	0.01	0.01	0.01	-0.01
Experience	0.99	1.00	-0.05	-0.03	-0.05	-0.05	0.01	-0.01	-0.00	0.01	0.01	0.01	-0.01



Nhận xét về các mối tương quan giữa các cột trong tập dữ liệu:

- Khách hàng có tuổi càng cao thì họ càng có nhiều năm kinh nghiệm trong công việc của mình
- Khách hàng mà có mức chi tiêu trung bình hàng tháng cao, tất nhiên họ phải có mức thu nhập hàng năm phải cao
- Khách hàng có chứng nhận tài khoản tiền gửi (CD) tại ngân hàng có khả năng sử dụng thẻ tín dụng do ngân hàng phát hành hoặc có tài khoản chứng khoán với ngân hàng
- Thu nhập hàng năm, mức chi tiêu hàng tháng, tài khoản chứng khoán với ngân hàng của khách hàng có liên quan đến việc khách hàng có chấp nhận khoản vay cá nhân được cung cấp trong dịch vụ trước hay không.

## 2.4. Kiểm tra Describe các biến có giá trị là số

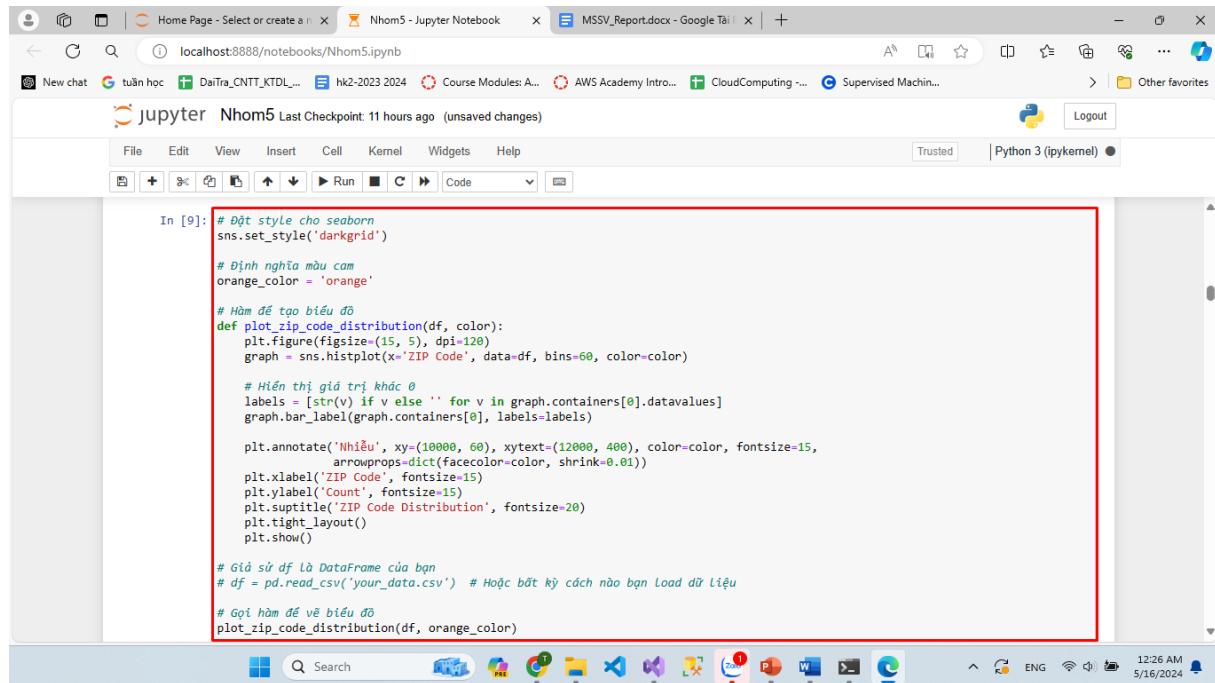
	count	mean	std	min	25%	50%	75%	max
Age	5000.0	45.338400	11.463166	23.0	35.0	45.0	55.0	67.0
Experience	5000.0	20.104600	11.467954	-3.0	10.0	20.0	30.0	43.0
Income	5000.0	73.774200	46.033729	8.0	39.0	64.0	98.0	224.0
ZIP Code	5000.0	93152.503000	2121.852197	9307.0	91911.0	93437.0	94608.0	96651.0
Family	5000.0	2.396400	1.147663	1.0	1.0	2.0	3.0	4.0
CCAvg	5000.0	1.937938	1.747659	0.0	0.7	1.5	2.5	10.0
Education	5000.0	1.881000	0.839669	1.0	1.0	2.0	3.0	3.0
Mortgage	5000.0	56.498800	101.713802	0.0	0.0	0.0	101.0	635.0
Personal Loan	5000.0	0.096000	0.294621	0.0	0.0	0.0	0.0	1.0
Securities Account	5000.0	0.104400	0.305809	0.0	0.0	0.0	0.0	1.0
CD Account	5000.0	0.060400	0.238250	0.0	0.0	0.0	0.0	1.0
Online	5000.0	0.596800	0.490589	0.0	0.0	1.0	1.0	1.0
CreditCard	5000.0	0.294000	0.455637	0.0	0.0	0.0	1.0	1.0

Nhận xét: Bộ dữ liệu này bao gồm nhiều thông tin khác nhau về độ tuổi, kinh nghiệm, thu nhập, tình trạng gia đình, chi tiêu thẻ tín dụng, trình độ học vấn, và các loại tài khoản ngân hàng. Các giá trị phân vị và độ lệch chuẩn cung cấp cái nhìn tổng quan về sự phân phối của các biến này.

## 2.5. Tiết xử lý dữ liệu

### 2.5.1. Xử lý nhiễu

#### 2.5.1.1. Cột Zip Code



```
In [9]: # Đặt style cho seaborn
sns.set_style('darkgrid')

# Định nghĩa màu cam
orange_color = 'orange'

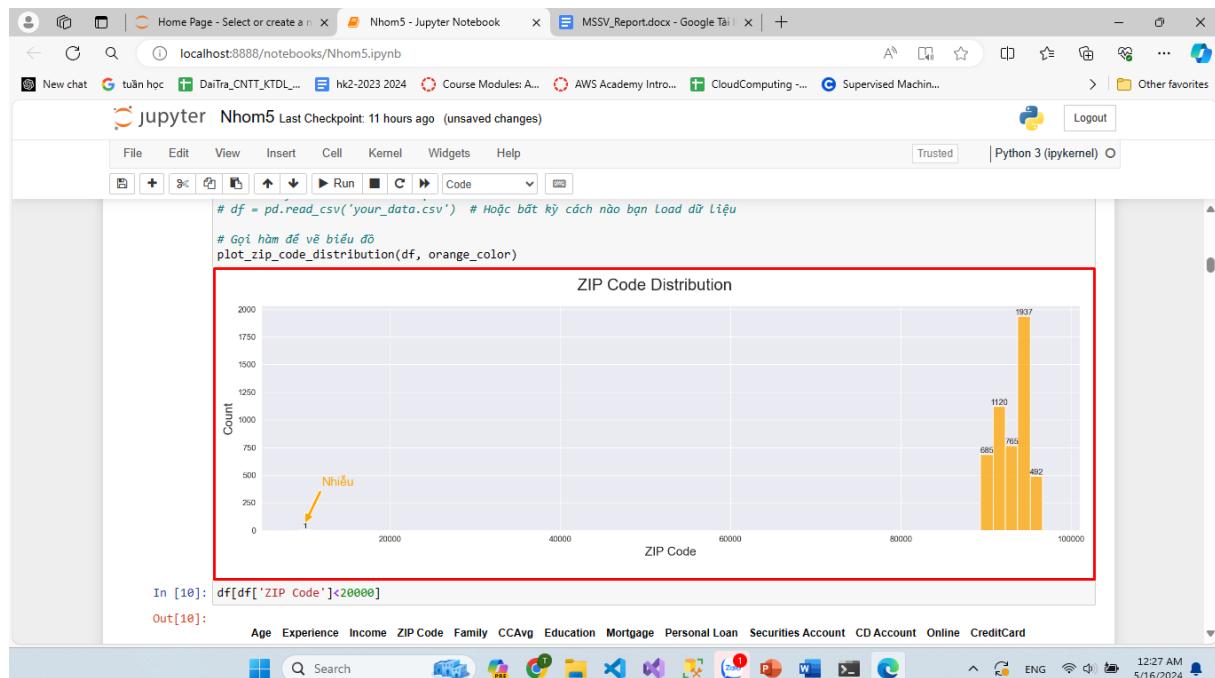
# Hàm để tạo biểu đồ
def plot_zip_code_distribution(df, color):
    plt.figure(figsize=(15, 5), dpi=120)
    graph = sns.histplot(x="ZIP Code", data=df, bins=60, color=color)

    # Hiển thị giá trị khác 0
    labels = [str(v) if v else '' for v in graph.containers[0].datavalues]
    graph.bar_label(graph.containers[0], labels=labels)

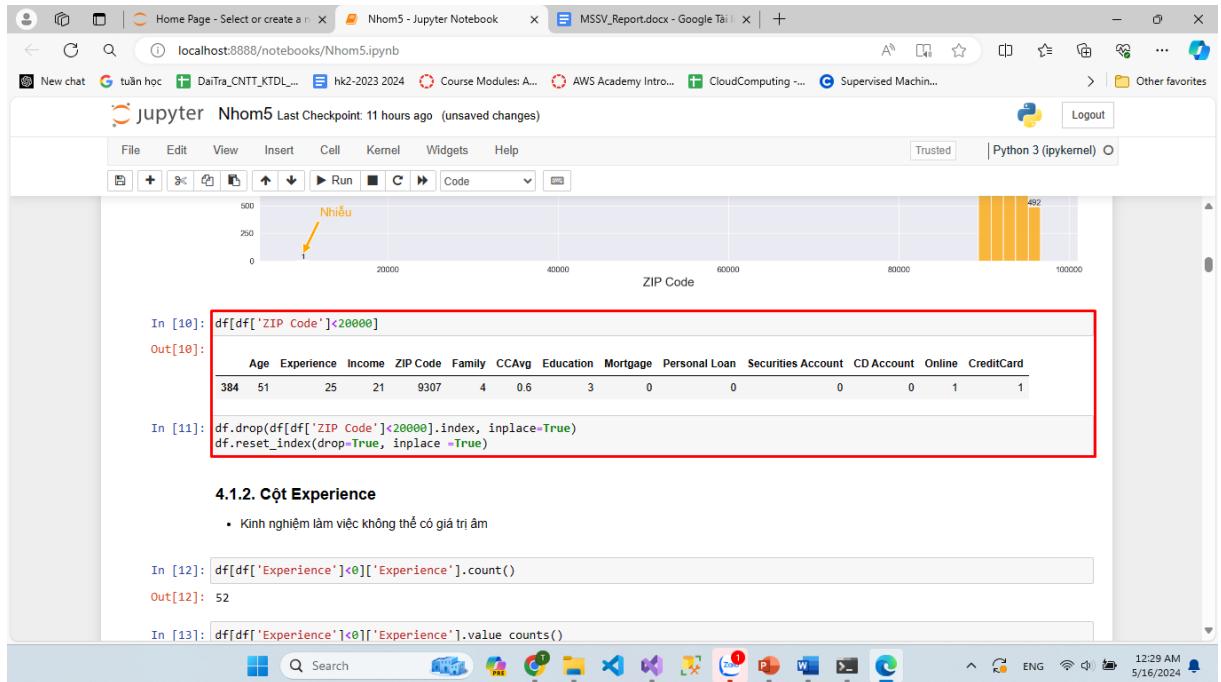
    plt.annotate('Nhiều', xy=(10000, 60), xytext=(12000, 400), color=color, fontsize=15,
                arrowprops=dict(facecolor=color, shrink=0.01))
    plt.xlabel('ZIP Code', fontsize=15)
    plt.ylabel('Count', fontsize=15)
    plt.suptitle('ZIP Code Distribution', fontsize=20)
    plt.tight_layout()
    plt.show()

# Giả sử df là Dataframe của bạn
# df = pd.read_csv('your_data.csv') # Hoặc bất kỳ cách nào bạn Load dữ liệu

# Gọi hàm để vẽ biểu đồ
plot_zip_code_distribution(df, orange_color)
```

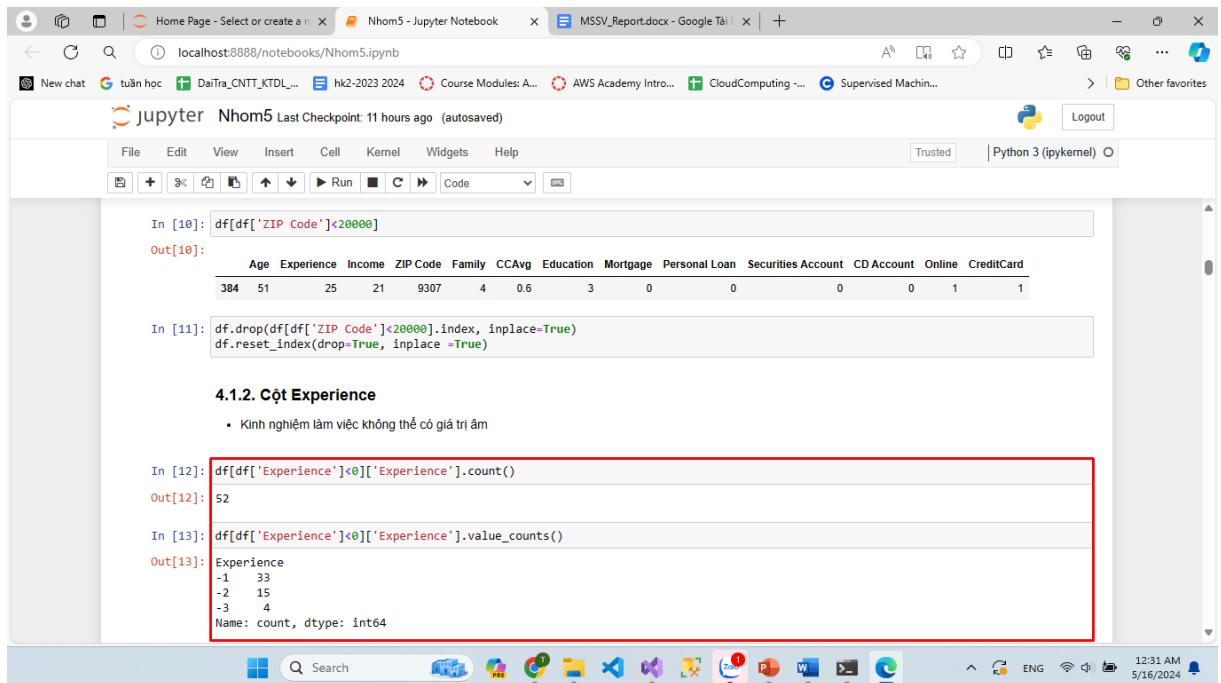


Ta thấy rằng có tồn tại một giá trị nhiễu ở cột này, tiến hành tìm kiếm và loại bỏ giá trị này.



### 2.5.1.2. Cột Experience

Kinh nghiệm là việc của một người không thể nào có giá trị âm, nên chúng ta sẽ tiến hành kiểm tra cột này.



Ta thấy có 52 giá trị âm trong cột **Experience**, với số lượng không đáng kể ( $[-1:32, -2:15, -3:4]$ ), có thể trong quá trình tạo dựng tập dữ liệu đã có sự sai sót.

Chúng ta có thể thay thế các giá trị nhiễu này thành giá trị tuyệt đối từ giá trị gốc ban đầu và kiểm tra lại.

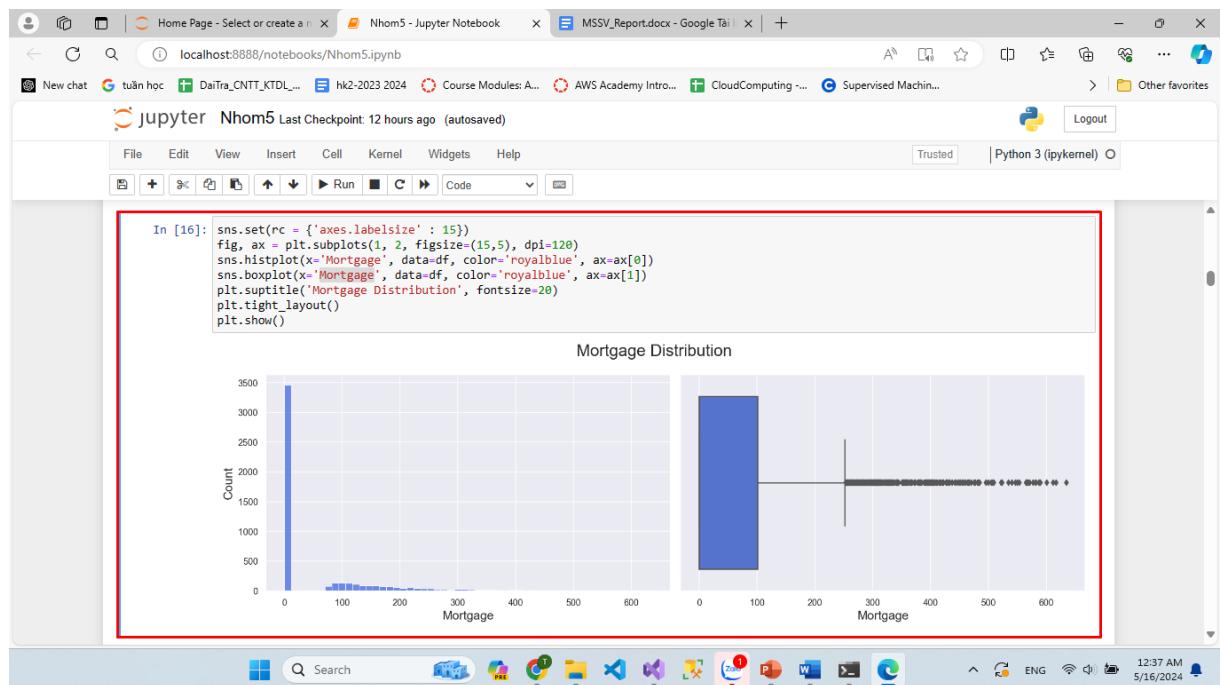
The screenshot shows a Jupyter Notebook interface with several code cells and their outputs:

- In [12]: `df[df['Experience'].isnull()]['Experience'].count()`  
Out[12]: 52
- In [13]: `df[df['Experience'].isnull()]['Experience'].value_counts()`  
Out[13]: Experience  
-1 33  
-2 15  
-3 4  
Name: count, dtype: int64
  - Ta thấy số lượng các giá trị nhiễu trong cột Experience rất nhỏ so với tập dữ liệu => có thể nó đã bị nhập nhầm
  - => Tiến hành thay thế các giá trị nhiễu này thành giá trị tuyệt đối
- In [14]: `df['Experience'] = df['Experience'].apply(abs)`
  - Kiểm tra lại
- In [15]: `df[df['Experience'].isnull()]['Experience'].count()`  
Out[15]: 0

A red box highlights the code in In [14] and its output in Out[15].

### 2.5.2. Xử lý dữ liệu ngoại lai (Outlier)

Chúng ta tiến hành xem xét sự phân bố các giá trị trong cột **Mortgage**



Có vẻ như cột **Mortgage** tồn tại khá nhiều các giá trị ngoại lai, chúng ta cần phải kiểm tra và xử lý các điểm dữ liệu ngoại lai thông qua phương pháp **Z-score**, với điều kiện  $Z\text{-score} > 3$ .

Z-score là một cách để đo lường độ lệch của một giá trị so với giá trị trung bình của tập dữ liệu, tính theo đơn vị độ lệch chuẩn (standard deviation).

Công thức tính Z-score cho một giá trị  $x$  là:

$$Z = \frac{x - \mu}{\sigma}$$

Trong đó:

- $x$  là giá trị cần tính
- $\mu$  là giá trị trung bình của tập dữ liệu
- $\sigma$  là độ lệch chuẩn của tập dữ liệu

Theo quy tắc ba-sigma trong thống kê, đối với một phân phối chuẩn (normal distribution):

- Khoảng  $\pm 1$  độ lệch chuẩn chứa khoảng 68% các giá trị.
- Khoảng  $\pm 2$  độ lệch chuẩn chứa khoảng 95% các giá trị.
- Khoảng  $\pm 3$  độ lệch chuẩn chứa khoảng 99.7% các giá trị.

Do đó, một giá trị có  $Z\text{-score} > 3$  hoặc  $< -3$  nằm ngoài 99.7% các giá trị thông thường, nên nó được coi là bất thường hoặc ngoại lệ.

In [17]: df[stats.zscore(df['Mortgage']) > 3]['Mortgage'].count()

Out[17]: 105

- Có 105 điểm dữ liệu có giá trị Z-score > 3. Do đó, có thể coi 105 bản ghi này là Outlier và loại bỏ những bản ghi này khỏi tập dữ liệu

In [18]: outlier\_indexes = df[stats.zscore(df['Mortgage']) > 3].index  
df.drop(outlier\_indexes, inplace=True)  
df.reset\_index(drop=True, inplace=True)

### 4.3. Xử lý dữ liệu bị thiếu (NULL)

In [19]: df.isnull().sum() + df.isna().sum()

Out[19]: Age 0

Có đến 105 giá trị Outlier trong cột **Mortgage**, tiến hành loại bỏ các bản ghi này ra khỏi tập dữ liệu.

#### 2.4.3. Xử lý dữ liệu bị thiếu (NULL)

### 4.3. Xử lý dữ liệu bị thiếu (NULL)

In [19]: df.isnull().sum() + df.isna().sum()

Out[19]:

Age	0
Experience	0
Income	0
ZIP Code	0
Family	0
CCAvg	0
Education	0
Mortgage	0
Personal Loan	0
Securities Account	0
CD Account	0
Online	0
CreditCard	0
dtype: int64	

- Có thể thấy, tập dữ liệu không có cột dữ liệu nào bị thiếu => clean dataset

### 4.4. Xử lý giá trị trùng lặp (Duplicate)

In [20]: # Đếm số dòng dữ liệu trùng lặp

Như đã biết, tập dữ liệu không có cột nào bị thiếu dữ liệu.

#### 2.5.4. Xử lý các giá trị trùng lặp (Duplicated)

In [20]:

```
# Đếm số dòng dữ liệu trùng lặp
num_duplicates = df.duplicated().sum()
print("Số dòng dữ liệu trùng lặp là:", num_duplicates)
```

Số dòng dữ liệu trùng lặp là: 0

- Như vậy, tập dataset không có các dòng dữ liệu trùng lặp nào

In [21]:

```
df['CCAvg'] = df['CCAvg']*12
```

12:54 AM 5/16/2024

Tập dữ liệu không có dòng nào bị trùng lặp dữ liệu với nhau.

### 2.5.5. Biến đổi đặc trưng

Trong tập dữ liệu, có cột CCAvg là Trung bình chi tiêu bằng thẻ tín dụng hàng tháng, cột Income là Thu nhập hàng năm của khách hàng, nên chúng ta tiến hành biến đổi cột CCAvg để có cùng miền giá trị với Income bằng cách nhân cho 12 tháng.

In [20]:

```
# Đếm số dòng dữ liệu trùng lặp
num_duplicates = df.duplicated().sum()
print("Số dòng dữ liệu trùng lặp là:", num_duplicates)
```

Số dòng dữ liệu trùng lặp là: 0

- Như vậy, tập dataset không có các dòng dữ liệu trùng lặp nào

In [21]:

```
df['CCAvg'] = df['CCAvg']*12
```

12:57 AM 5/16/2024

## 5. Phân tích dữ liệu thăm dò (EDA) và Trực quan hóa

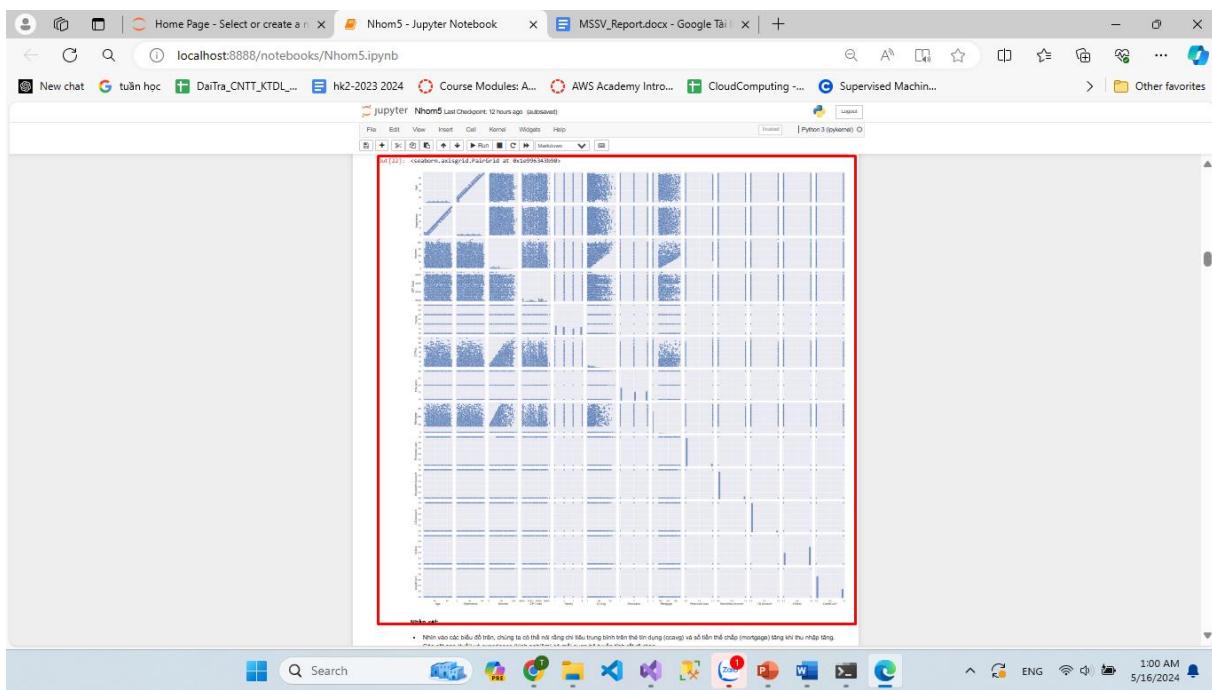
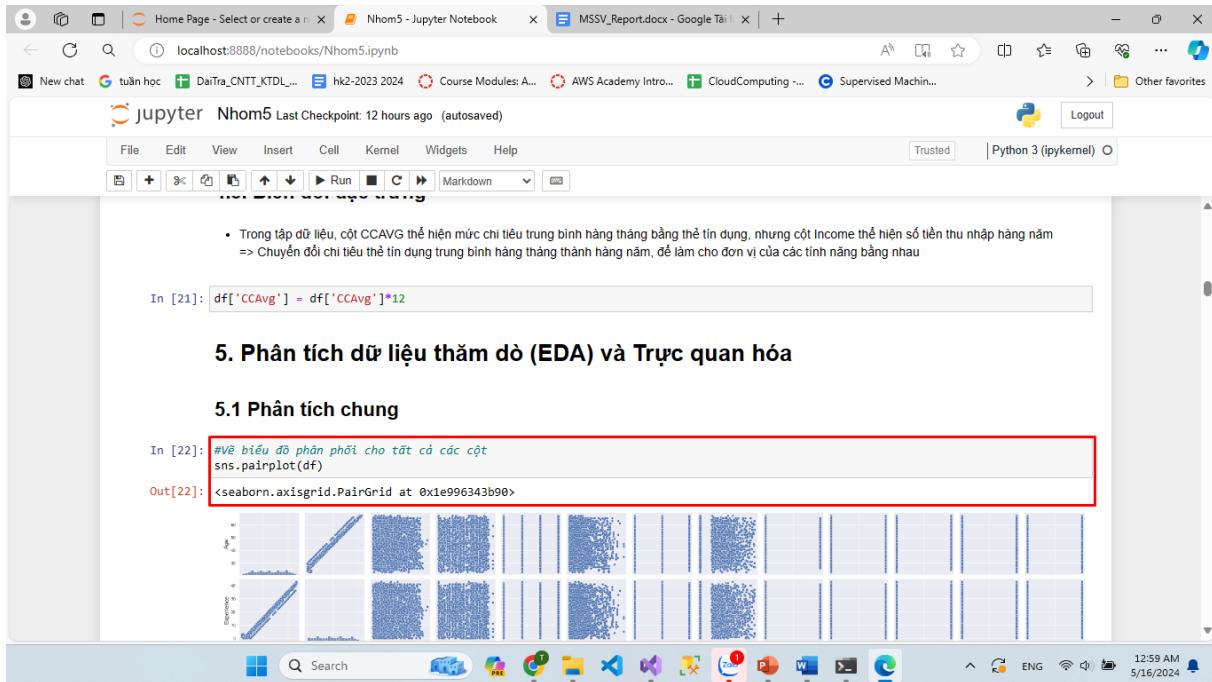
### 5.1 Phân tích chung

```
In [22]: # Vẽ biểu đồ phân phối cho tất cả các cột
sns.pairplot(df)
```

```
Out[22]: <seaborn.axisgrid.PairGrid at 0x1e996343b90>
```

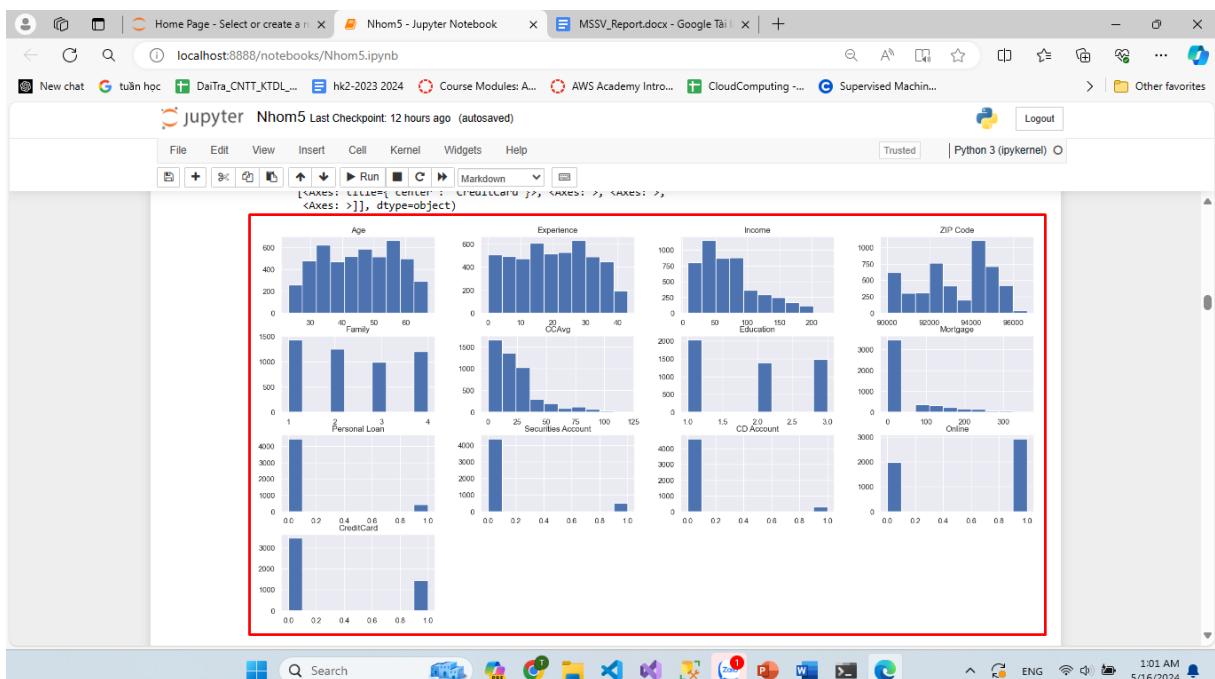
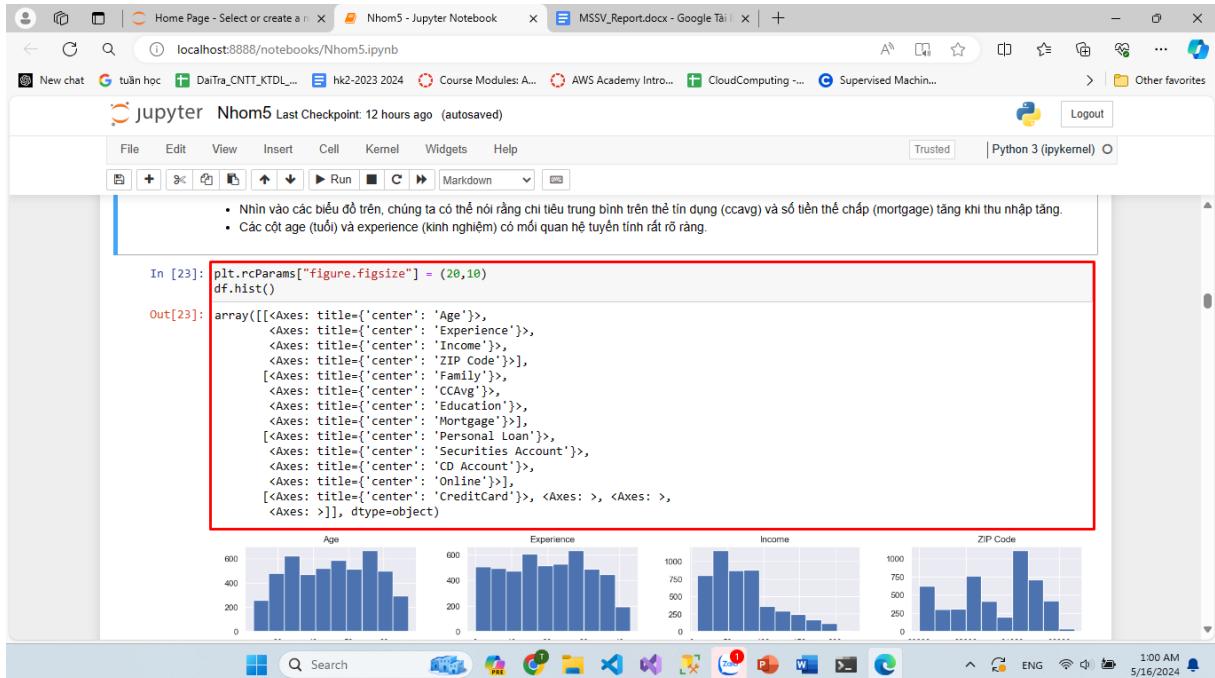
## 2.6. Phân tích dữ liệu thăm dò (EDA) và Trực quan hóa

### 2.6.1. Phân tích chung



- Nhìn vào các biểu đồ trên, chúng ta có thể nói rằng chi tiêu trung bình trên thẻ tín dụng (ccavg) và số tiền thẻ chấp (mortgage) tăng khi thu nhập tăng.

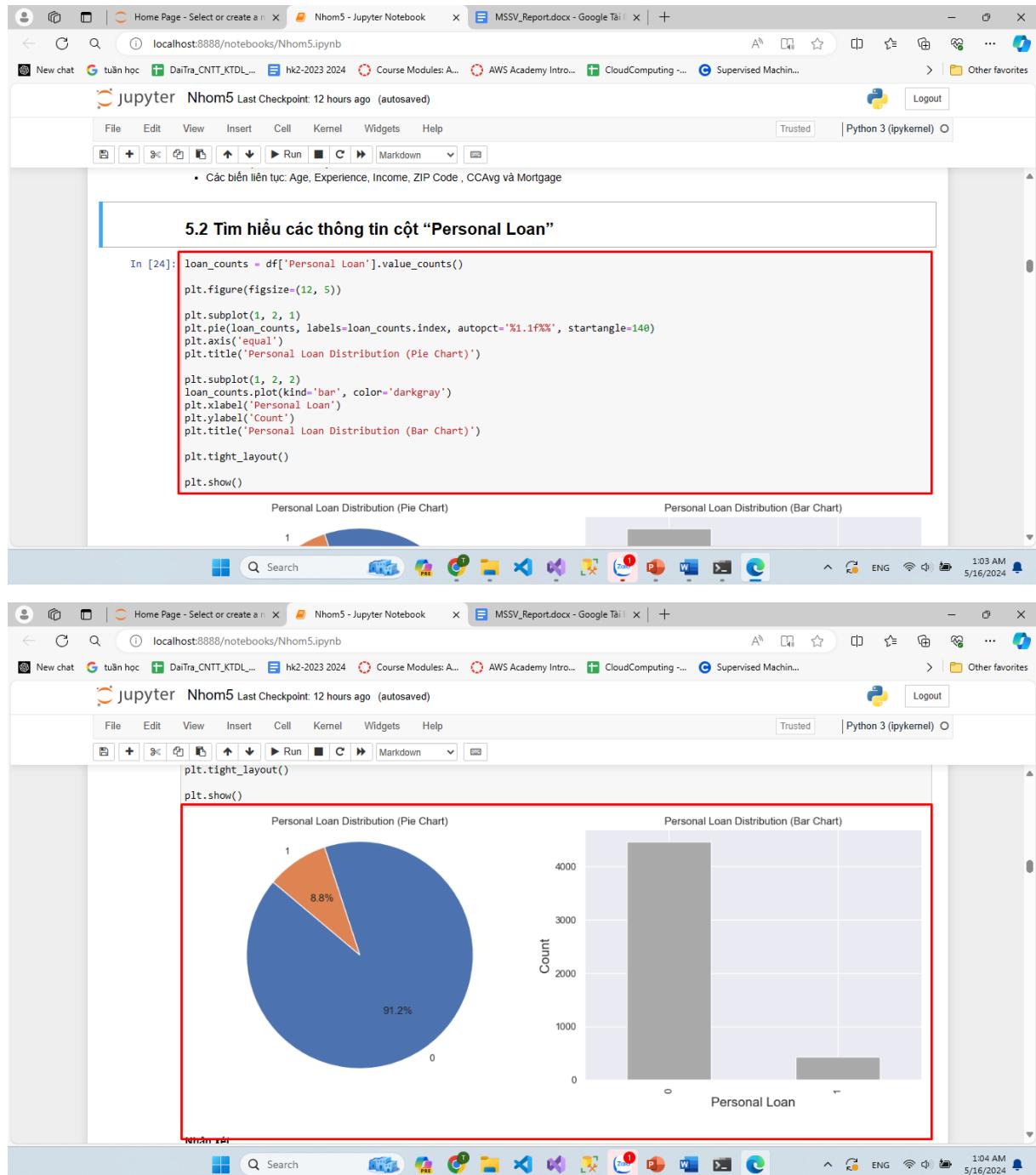
- Các cột age (tuổi) và experience (kinh nghiệm) có mối quan hệ tuyến tính rất rõ ràng.



Các cột trong tập dữ liệu được chia thành 2 loại:

- Các biến phân loại: Family, Education, Personal Loan, Securities Account, CD Account, Online và CreditCard.
- Các biến liên tục: Age, Experience, Income, ZIP Code , CCAvg và Mortgage.

## 2.6.2. Tìm hiểu cột Personal Loan



- Biểu đồ cho thấy phần lớn (91.2%) khách hàng không có khoản vay cá nhân, trong khi chỉ có một phần nhỏ là có khoản vay.
- Sự phân bổ giữa 2 lớp phân loại Personal Loan là không đồng đều.

## 2.6.3. Tìm hiểu các thông tin liên quan/ảnh hưởng đến cột Personal Loan

### 2.6.3.1. Phân tích thông tin cá nhân khách hàng: Age, Experience, và Income

```

In [25]: def draw_boxplots(df, column):
    mean = df[column].mean()
    ax.axvline(mean, color='r', linestyle='--', label=f'Mean: {mean}')
    ax.legend()

fig, axes = plt.subplots(3, 3, figsize=(18, 18))

# Biểu đồ Age
sns.boxplot(df["Age"], ax=axes[0][0], color="mediumslateblue")
axes[0][0].set(xlabel="Distribution of Age")

pp = sns.histplot(df["Age"], ax=axes[0][1], bins=10, color="mediumslateblue")
axes[0][1].set_title("Histogram of Age")
axes[0][1].set_xlabel("Age")
axes[0][1].set_ylabel("Density")

sns.kdeplot(data=df, x="Age", hue="Personal Loan", fill=True, common_norm=False, palette="muted", ax=axes[0][2])
axes[0][2].set_title("Density Plot of Age by Personal Loan")
axes[0][2].set_xlabel("Age")
axes[0][2].set_ylabel("Density")

# Biểu đồ Experience
sns.boxplot(df["Experience"], ax=axes[1][0], color="mediumslateblue")
axes[1][0].set(xlabel="Distribution of Experience")

pp = sns.histplot(df["Experience"], ax=axes[1][1], bins=10, color="mediumslateblue")
axes[1][1].set_title("Histogram of Experience")
axes[1][1].set_xlabel("Experience")
draw_kde(pp, "Experience")

sns.kdeplot(data=df, x="Experience", hue="Personal Loan", fill=True, common_norm=False, palette="muted", ax=axes[1][2])
axes[1][2].set_title("Density Plot of Experience by Personal Loan")
axes[1][2].set_xlabel("Experience")
axes[1][2].set_ylabel("Density")

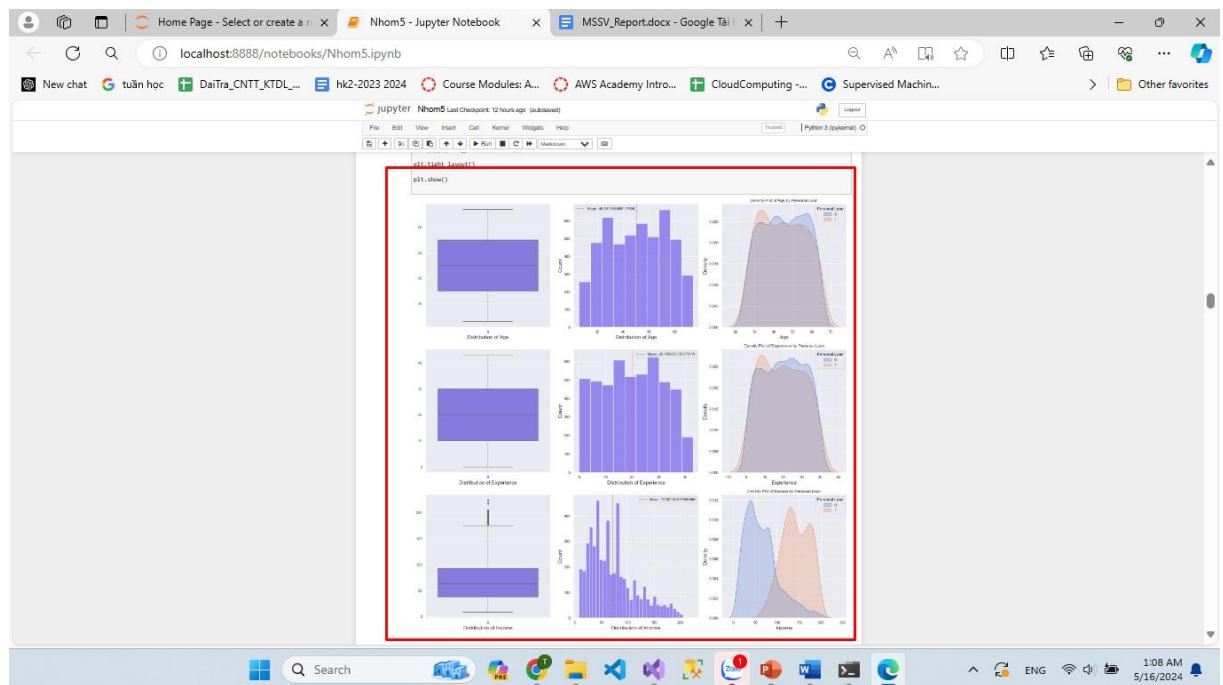
# Biểu đồ Income
sns.boxplot(df["Income"], ax=axes[2][0], color="mediumslateblue")
axes[2][0].set(xlabel="Distribution of Income")

pp = sns.histplot(df["Income"], ax=axes[2][1], color="mediumslateblue")
axes[2][1].set_title("Histogram of Income")
axes[2][1].set_xlabel("Income")
draw_kde(pp, "Income")

sns.kdeplot(data=df, x="Income", hue="Personal Loan", fill=True, common_norm=False, palette="muted", ax=axes[2][2])
axes[2][2].set_title("Density Plot of Income by Personal Loan")
axes[2][2].set_xlabel("Income")
axes[2][2].set_ylabel("Density")

plt.tight_layout()

```



Nhận xét 1:

#### 1. Tuổi (Age):

- Biểu đồ boxplot cho thấy phân phối rất đồng đều xung quanh giá trị trung bình và không có các điểm ngoại lai.
- Biểu đồ histogram cho thấy có một phân phối đối xứng và giá trị trung bình cũng như Q2 gần nhau.

- Có một số đỉnh nhỏ tại các vị trí của Q1, Q2 và Q3.

2. Kinh nghiệm (Experience):

- Phân phối rất tương tự như của Age.
- Giá trị trung bình và Q2 giống nhau. Có một phân phối đối xứng.

3. Thu nhập (Income):

- Biểu đồ boxplot cho thấy một số điểm ngoại lai vượt qua ngưỡng trên.
- Biểu đồ histogram cho thấy phân phối hướng phải phân phối bên phải.

Nhận xét 2:

1. Biểu Đồ Mật Độ Tuổi Theo Khoản Vay Cá Nhân:

- Có hai phân phối chồng lên nhau, một cho những người có khoản vay cá nhân (màu cam) và một cho những người không có (màu xanh).
- Cả hai đều đạt đỉnh xung quanh độ tuổi 40, nhưng phân phối cho những người có khoản vay cá nhân có xu hướng dịch chuyển sang phải, cho thấy độ tuổi cao hơn.

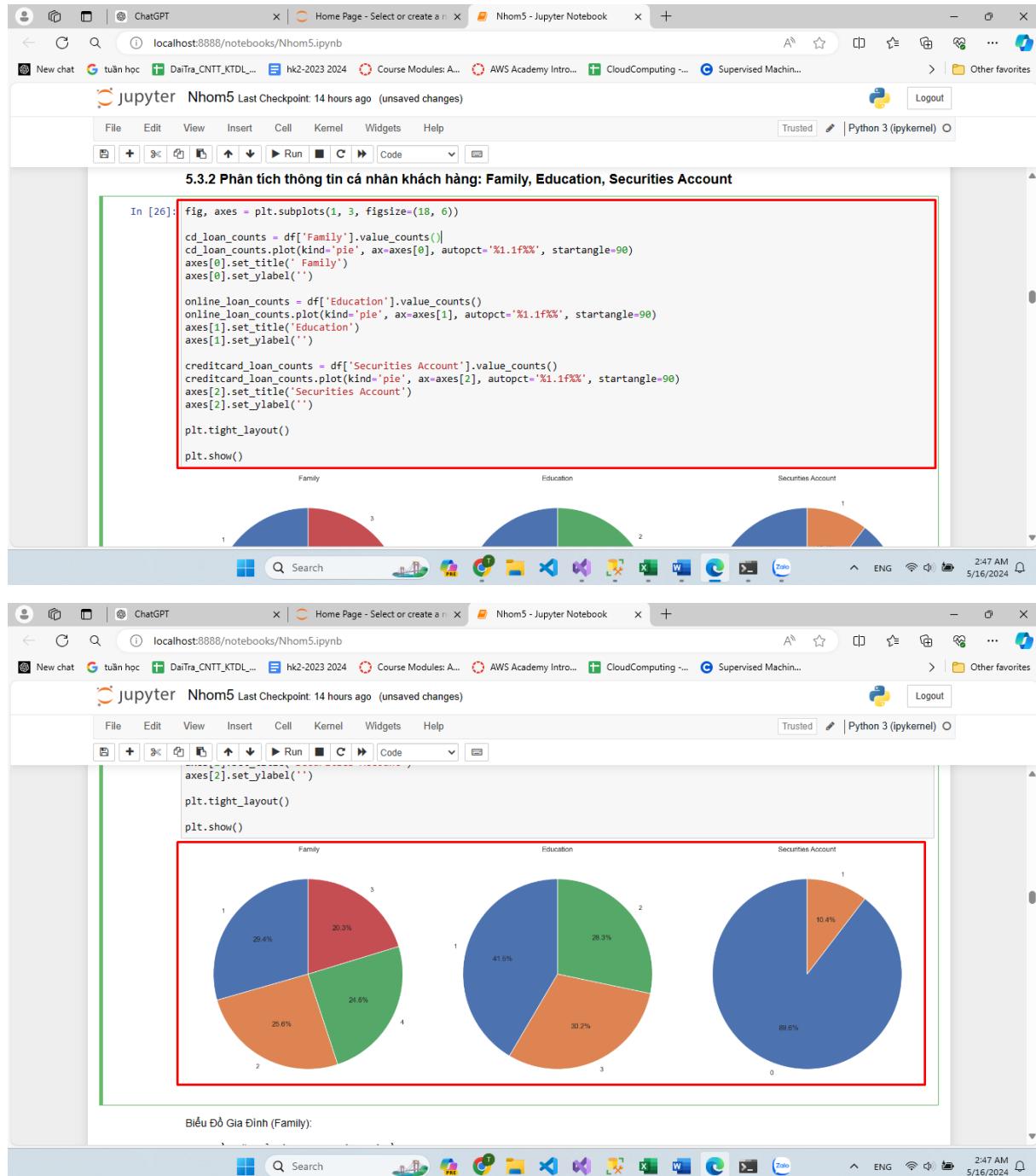
2. Biểu Đồ Mật Độ Kinh Nghiệm Theo Khoản Vay Cá Nhân:

- Hai phân phối gần như giống nhau, đều đạt đỉnh ở khoảng 20 năm kinh nghiệm.

3. Biểu Đồ Mật Độ Thu Nhập Theo Khoản Vay Cá Nhân:

- Có sự phân biệt rõ ràng; những cá nhân có thu nhập cao hơn có xu hướng vay cá nhân, như được chỉ ra bởi phân phối màu cam đạt đỉnh ở mức thu nhập cao hơn.
- Nhìn chung, biểu đồ cho thấy rằng tuổi tác và thu nhập có thể là những yếu tố quan trọng ảnh hưởng đến quyết định vay cá nhân, trong khi kinh nghiệm không cho thấy sự khác biệt đáng kể giữa hai nhóm. Điều này có thể hữu ích cho các tổ chức tài chính khi xem xét hồ sơ vay và chiến lược tiếp thị của họ.

### 2.6.3.2. Phân tích thông tin cá nhân khách hàng: Family, Education, Securities Account



#### 1. Biểu Đồ Gia Đình (Family):

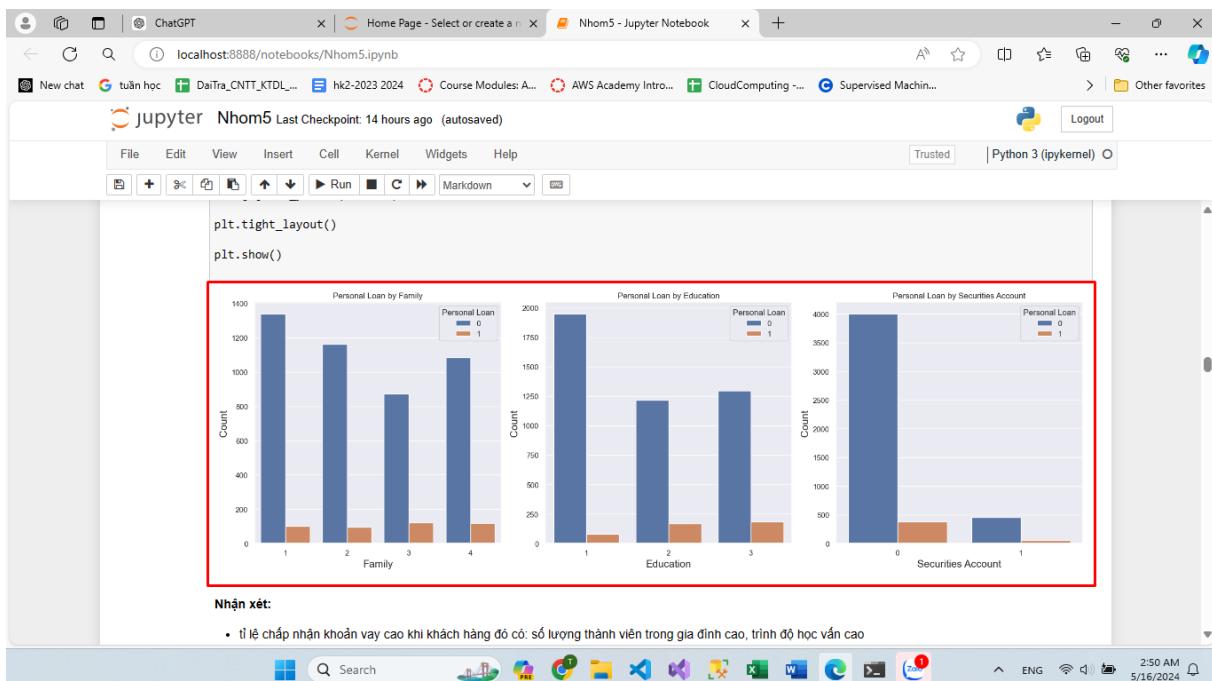
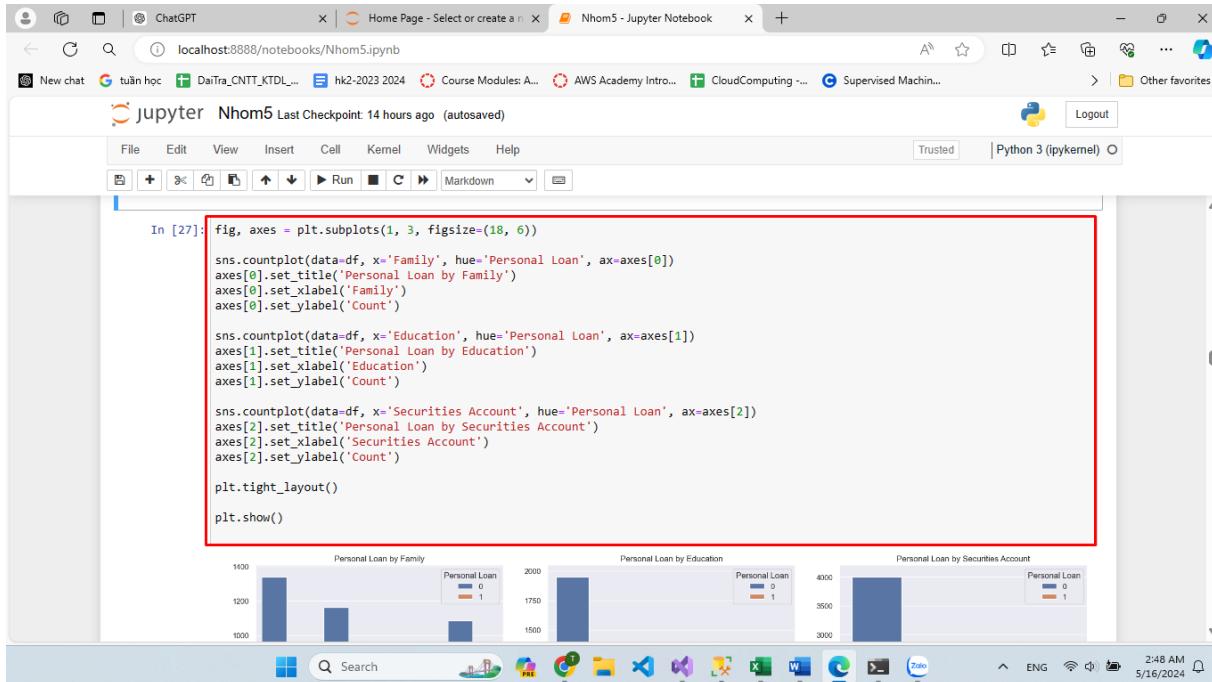
- Phân trăm của từng loại gia đình khá đều.
- Phân đoạn màu xanh (29.4%) có vẻ là lớn nhất, cho thấy đó là loại gia đình phổ biến nhất trong dữ liệu.

#### 2. Biểu Đồ Học Vấn (Education):

- Phân đoạn màu xanh chiếm ưu thế lớn (41.5%), có thể chỉ ra rằng đa số mọi người trong dữ liệu có trình độ học vấn này.

### 3. Biểu Đồ Tài Khoản chứng khoán (Securities Account):

- Phân đoạn màu xanh lớn (89.6%) cho thấy rằng đa số mọi người không sở hữu tài khoản chứng khoán.



Nhận xét:

- Tỉ lệ chấp nhận khoản vay cao khi khách hàng đó có: số lượng thành viên trong gia đình cao, trình độ học vấn cao
- Thuộc tính có tài khoản chứng khoán có vẻ không có tương quan với tỉ lệ chấp nhận khoản vay

```
In [28]: fig, axes = plt.subplots(1, 3, figsize=(18, 6))

cd_loan_counts = df['CD Account'].value_counts()
cd_loan_counts.plot(kind='pie', ax=axes[0], autopct='%1.1f%%', startangle=90)
axes[0].set_title(' CD Account')
axes[0].set_ylabel('')

online_loan_counts = df['Online'].value_counts()
online_loan_counts.plot(kind='pie', ax=axes[1], autopct='%1.1f%%', startangle=90)
axes[1].set_title(' by Online')
axes[1].set_ylabel('')

creditcard_loan_counts = df['CreditCard'].value_counts()
creditcard_loan_counts.plot(kind='pie', ax=axes[2], autopct='%1.1f%%', startangle=90)
axes[2].set_title('CreditCard')
axes[2].set_ylabel('')

plt.tight_layout()
plt.show()
```

The figure consists of three pie charts side-by-side. The first chart, titled 'CD Account', shows two segments: one large blue segment labeled '0' and one small orange segment labeled '1'. The orange segment is 5.8% of the total. The second chart, titled 'by Online', shows two segments: a large blue segment labeled '0' at 59.6% and a smaller orange segment labeled '1' at 40.4%. The third chart, titled 'CreditCard', shows two segments: a large blue segment labeled '0' at 70.5% and a smaller orange segment labeled '1' at 29.5%.

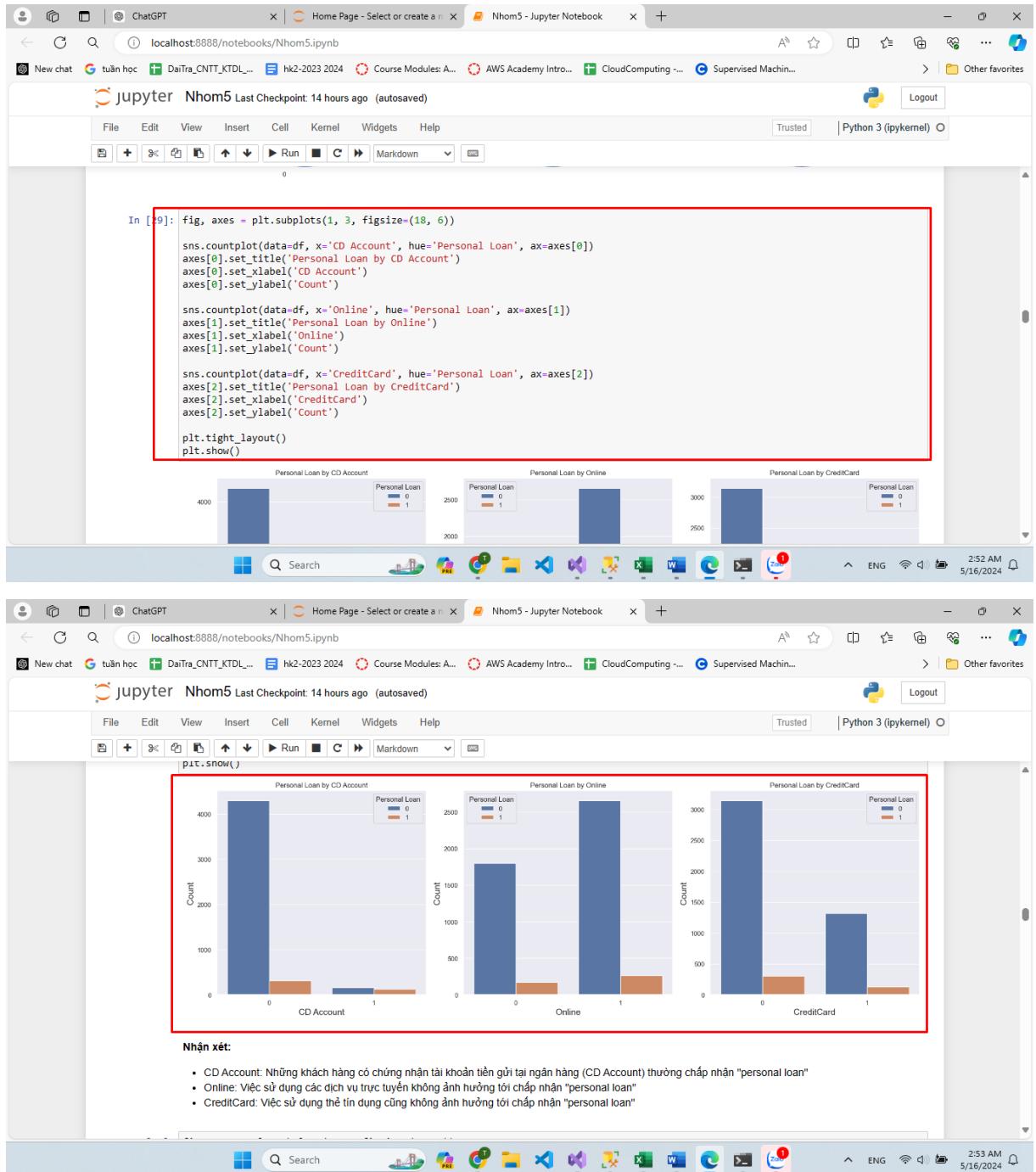
  

```
plt.tight_layout()
plt.show()
```

The figure consists of three pie charts side-by-side. The first chart, titled 'CD Account', shows two segments: one large blue segment labeled '0' and one small orange segment labeled '1'. The orange segment is 5.8% of the total. The second chart, titled 'by Online', shows two segments: a large blue segment labeled '0' at 59.6% and a smaller orange segment labeled '1' at 40.4%. The third chart, titled 'CreditCard', shows two segments: a large blue segment labeled '0' at 70.5% and a smaller orange segment labeled '1' at 29.5%.

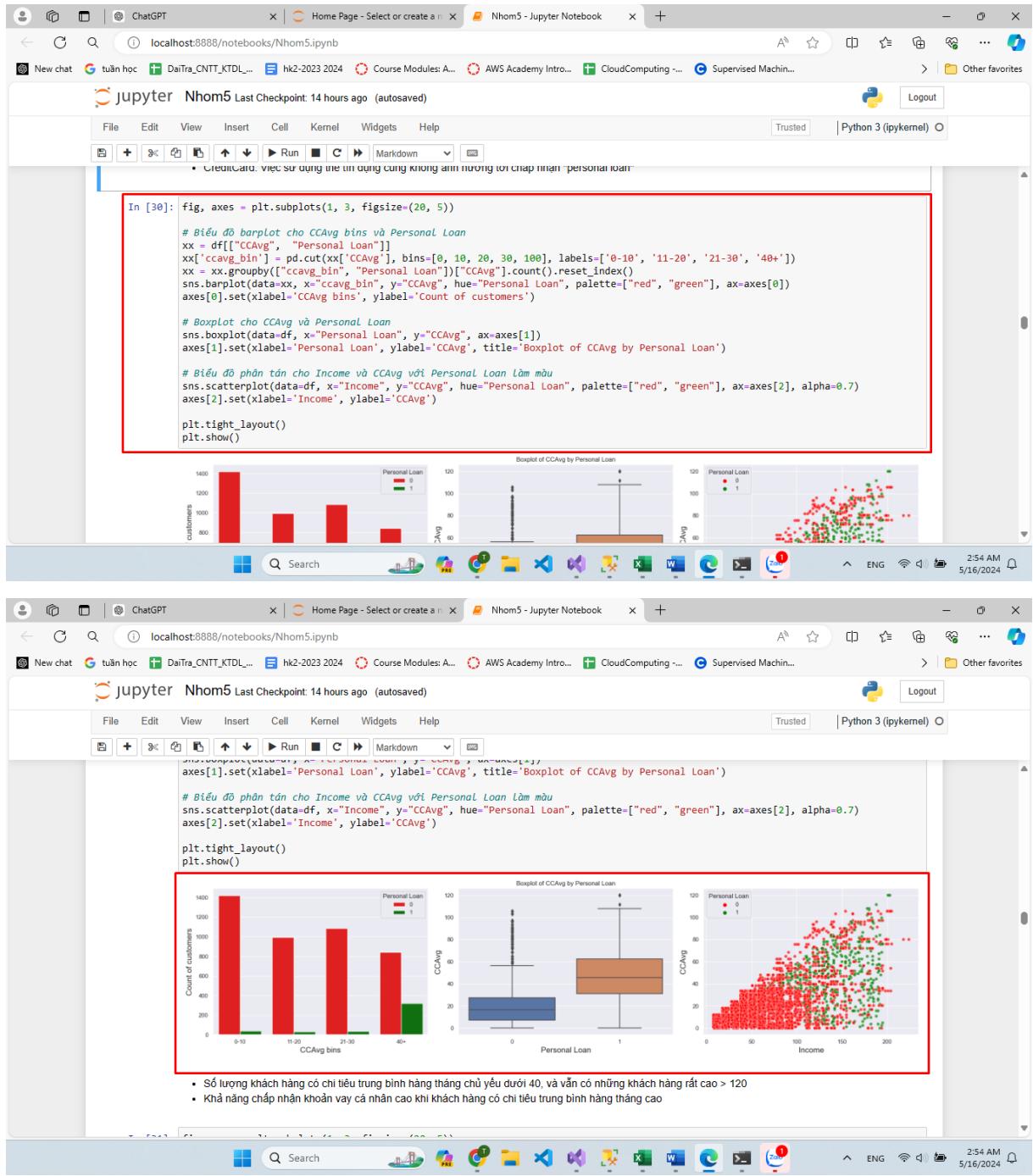
```
In [29]: fig, axes = plt.subplots(1, 3, figsize=(18, 6))

sns.countplot(data=df, x='CD Account', hue='Personal Loan', ax=axes[0])
axes[0].set_title('Personal Loan by CD Account')
```

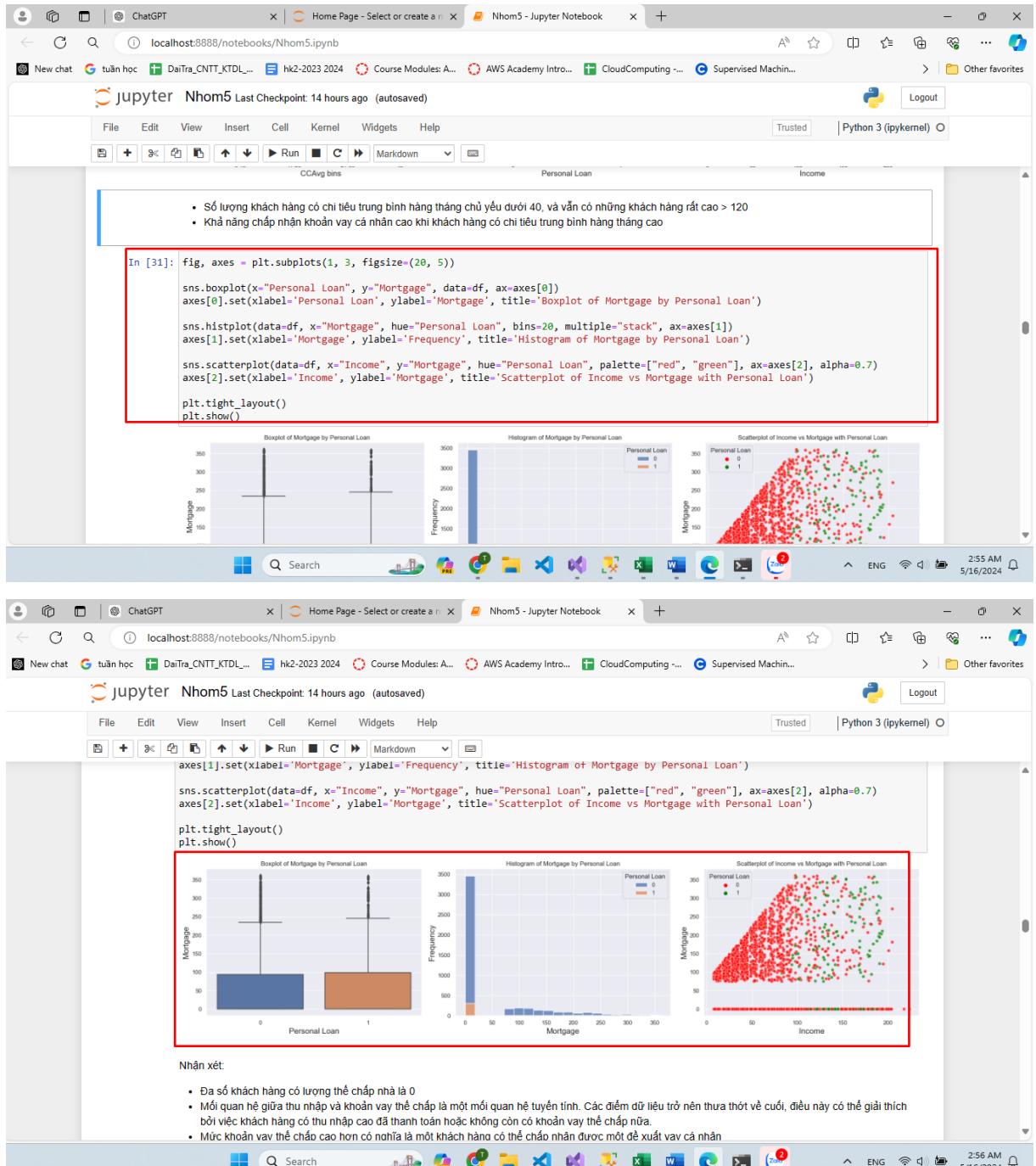


## Nhận xét:

- CD Account: Những khách hàng có chứng nhận tài khoản tiền gửi tại ngân hàng (CD Account) thường chấp nhận "personal loan"
- Online: Việc sử dụng các dịch vụ trực tuyến không ảnh hưởng tới chấp nhận "personal loan"
- CreditCard: Việc sử dụng thẻ tín dụng cũng không ảnh hưởng tới chấp nhận "personal loan"



- Số lượng khách hàng có chi tiêu trung bình hàng tháng chủ yếu dưới 40, và vẫn có những khách hàng rất cao > 120
- Khả năng chấp nhận khoản vay cá nhân cao khi khách hàng có chi tiêu trung bình hàng tháng cao



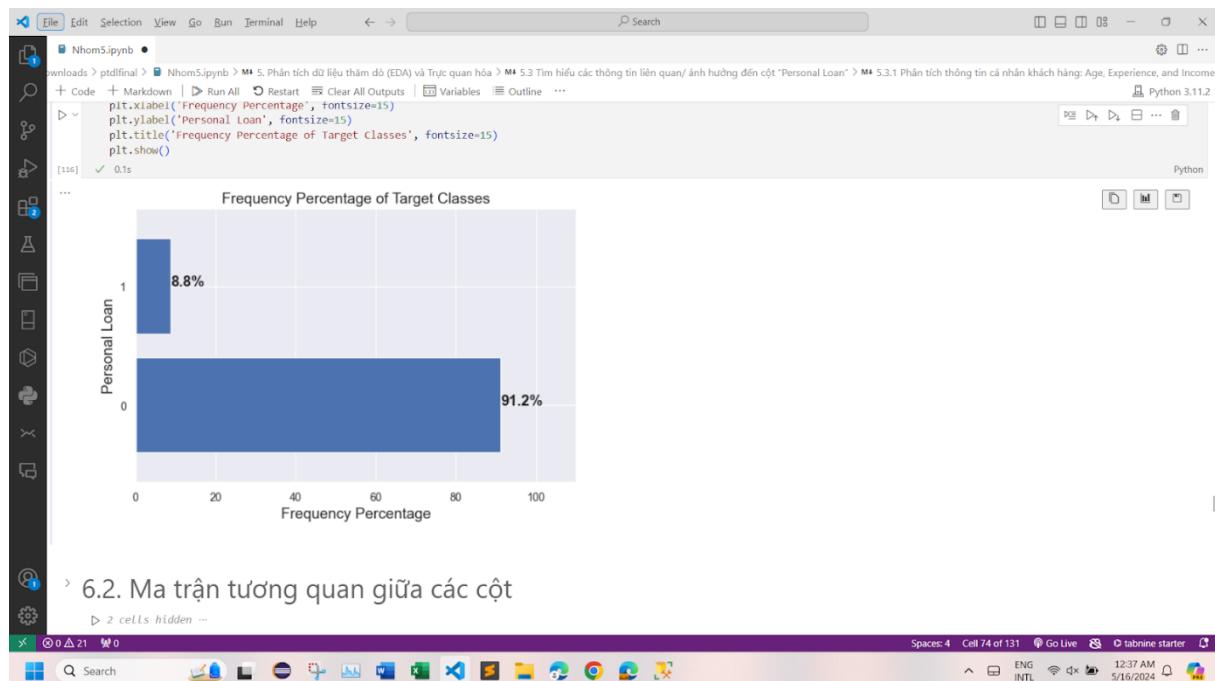
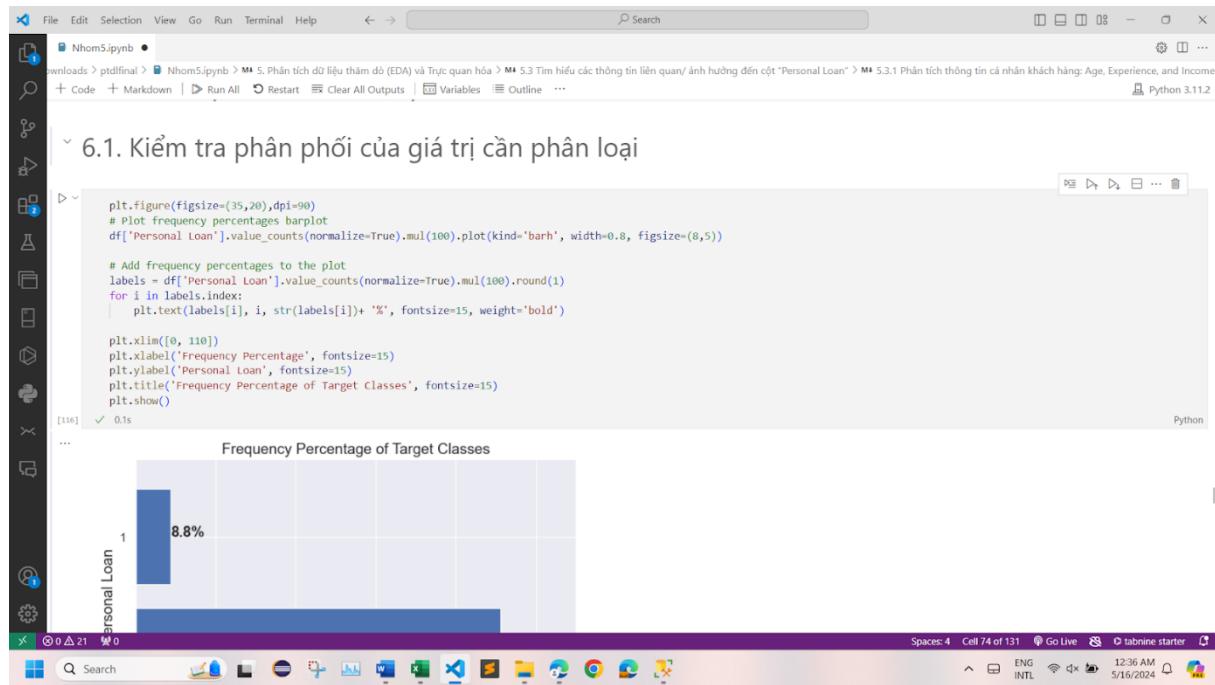
Nhận xét:

- Đa số khách hàng có lượng thẻ chấp nhà là 0
- Mối quan hệ giữa thu nhập và khoản vay thẻ chấp là một mối quan hệ tuyến tính. Các điểm dữ liệu trở nên thưa thớt về cuối, điều này có thể giải thích bởi việc khách hàng có thu nhập cao đã thanh toán hoặc không còn có khoản vay thẻ chấp nữa.
- Mức khoản vay thẻ chấp cao hơn có nghĩa là một khách hàng có thể chấp nhận được một mức vay cá nhân

- Mức khoản vay thế chấp cao hơn có nghĩa là một khách hàng có thể chấp nhận được một đề xuất vay cá nhân

## 2.7. Thiết lập các mô hình phân loại cho dự đoán “Personal Loan”

### 2.7.1. Kiểm tra phân phối của giá trị cần phân loại



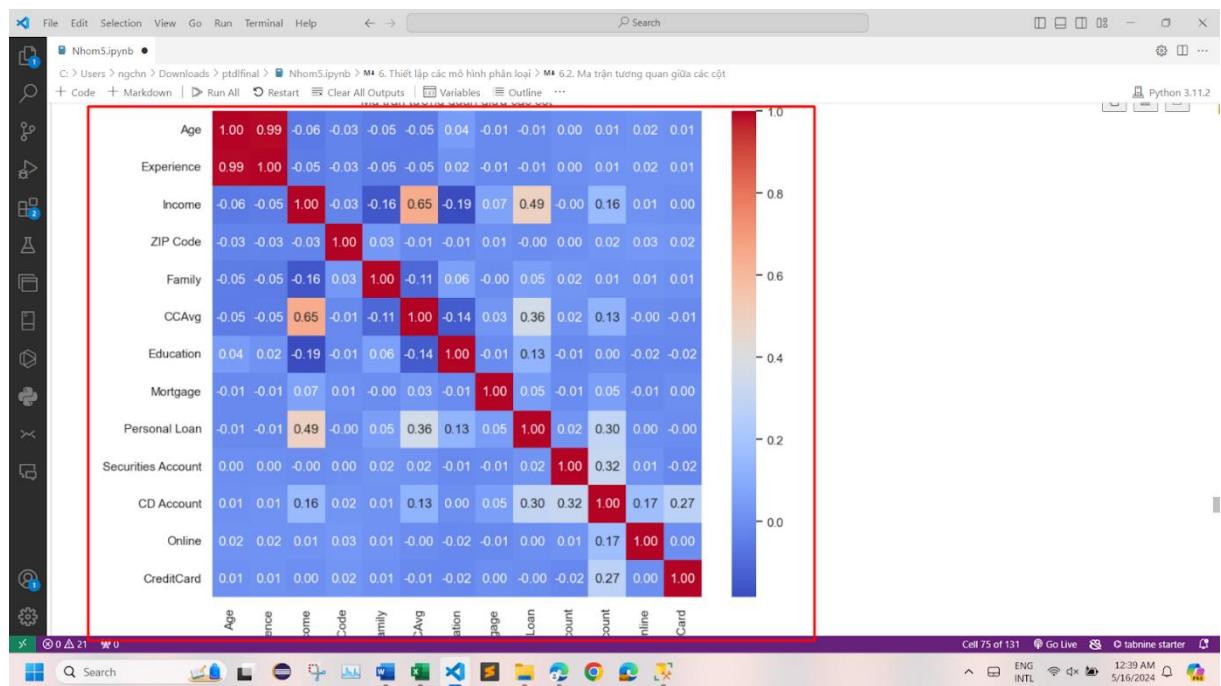
## 2.7.2. Ma trận tương quan giữa các cột

The screenshot shows a Jupyter Notebook interface with a Python code cell and its output. The code calculates the correlation matrix and generates a heatmap. The heatmap shows correlations between columns: Age, Experience, Income, ZIP Code, Family, CCAvg, and Education. The color scale ranges from -0.2 (blue) to 1.0 (red). The matrix is symmetric, with diagonal elements being 1.0.

```
# Tính toán ma trận tương quan
correlation_matrix = df.corr()

# Vẽ heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Ma trận tương quan giữa các cột')
plt.show()
```

	Age	Experience	Income	ZIP Code	Family	CCAvg	Education
Age	1.00	0.99	-0.06	-0.03	-0.05	-0.05	0.04
Experience	0.99	1.00	-0.05	-0.03	-0.05	-0.05	0.02
Income	-0.06	-0.05	1.00	-0.16	0.65	-0.19	0.07
ZIP Code	-0.03	-0.03	-0.03	1.00	0.03	-0.01	-0.01
Family	-0.05	-0.05	0.03	1.00	-0.11	0.06	-0.00
CCAvg	-0.05	-0.05	0.65	-0.01	-0.11	1.00	-0.14
Education	0.04	0.02	-0.19	-0.01	0.06	-0.14	1.00



Nhận xét:

- Ta thấy có 4 cột có mối tương quan cao so với là: **Income, CCAvg, Education, CD Account**

- Vậy có thể chia làm 2 trường hợp để tạo các mô hình phân loại **Personal Loan**

### 2.7.3. Phân chia dữ liệu thành 2 tập Train và Test

Trường hợp 1: Chỉ có các cột có mối tương quan cao: Income, CCAvg, Education, CD Account

The screenshot shows a Jupyter Notebook interface with the following content:

## 6.3. Phân chia dữ liệu thành 2 tập Train và Test

Trường hợp 1: Chỉ có các cột có mối tương quan cao: Income, CCAvg, Education, CD Account

```
selected_features = ['Income', 'CCAvg', 'Education', 'CD Account', 'Personal Loan']
df_selected = df[selected_features]

# Train-Test Split cho trường hợp 1
x_selected = df_selected.drop('Personal Loan', axis=1)
y_selected = df_selected['Personal Loan']

X_train_selected, X_test_selected, y_train_selected, y_test_selected = train_test_split(x_selected, y_selected, test_size=0.2, random_state=42)
```

[118] 0.0s

## Trường hợp 2: Lấy tất cả các cột

```
# Trường hợp 2: Tất cả các cột trong df
X_all = df.drop('Personal Loan', axis=1)
y_all = df['Personal Loan']

X_train_all, X_test_all, y_train_all, y_test_all = train_test_split(X_all, y_all, test_size=0.2, random_state=42)
```

[119] 0.0s

## 6.4. Decision Tree Model

Trường hợp 2: Lấy tất cả các cột

File Edit Selection View Go Run Terminal Help ← → Search

Nhom5.ipynb •

C:\Users\nghcn\Downloads>ptdfinal> Nhom5.ipynb > M4 6. Thiết lập các mô hình phân loại > M4 6.2. Ma trận tương quan giữa các cột

+ Code + Markdown ▶ Run All ⌘ Restart ⌘ Clear All Outputs Variables Outline ...

Python 3.11.2

## 6.3. Phân chia dữ liệu thành 2 tập Train và Test

Trường hợp 1: Chỉ có các cột có mối tương quan cao: Income, CCAvg, Education, CD Account

```
selected_features = ['Income', 'CCAvg', 'Education', 'CD Account', 'Personal Loan']
df_selected = df[selected_features]

# Train-Test Split cho trường hợp 1
X_selected = df_selected.drop('Personal Loan', axis=1)
y_selected = df_selected['Personal Loan']
X_train_selected, X_test_selected, y_train_selected, y_test_selected = train_test_split(X_selected, y_selected, test_size=0.2, random_state=42)
```

[118] ✓ 0.0s Python

Trường hợp 2: Lấy tất cả các cột

```
# Trường hợp 2: Tất cả các cột trong df
X_all = df.drop('Personal Loan', axis=1)
y_all = df['Personal Loan']
X_train_all, X_test_all, y_train_all, y_test_all = train_test_split(X_all, y_all, test_size=0.2, random_state=42)
```

## 6.4. Decision Tree Model

### Trường hợp 1:



#### 2.7.4. Decision Tree Model

#### *2.7.4.1. Trường hợp 1:*

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help, Search.
- Toolbar:** Home, New, Open, Save, Run Cell, Kernel, Help, Python 3.11.2.
- Code Cell:** Contains Python code for building a Decision Tree classifier, predicting on training and test sets, and calculating accuracy. The code is highlighted with a red box.
- Output Cell:** Displays the accuracy scores for both training and testing datasets.
- Status Bar:** Cell 75 of 131, Go Live, tabnine starter, ENG INTL, 12:50 AM, 5/16/2024.

```
# Xây dựng mô hình Decision Tree
dt_selected = DecisionTreeClassifier(random_state=42)
dt_selected.fit(X_train_selected, y_train_selected)

# Dự đoán trên tập huấn luyện
y_train_pred_selected = dt_selected.predict(X_train_selected)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_selected = accuracy_score(y_train_selected, y_train_pred_selected)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_selected)

# Dự đoán trên tập kiểm tra
y_test_pred_selected = dt_selected.predict(X_test_selected)

# Đánh giá hiệu suất
accuracy_test_selected = accuracy_score(y_test_selected, y_test_pred_selected)
print("Accuracy trên tập kiểm tra (trường hợp 1):", accuracy_test_selected)
print("Báo cáo phân loại (trường hợp 1):")
print(classification_report(y_test_selected, y_test_pred_selected))

[130] ✓ 0%
```

... Accuracy trên tập huấn luyện (trường hợp 1): 0.9984674329501916  
Accuracy trên tập kiểm tra (trường hợp 1): 0.9713993871297242  
Báo cáo phân loại (trường hợp 1):

Nhom5.ipynb

```
C:\Users\ngchh\Downloads\ptdfinal> Nhom5.ipynb > M4 6. Thiết lập các mô hình phân loại > M4 6.2. Ma trận tương quan giữa các cột
+ Code + Markdown | ▶ Run All ⚡ Restart ⌛ Clear All Outputs | ⌂ Variables ⌂ Outline ...
D> ...
print("Accuracy trên tập kiểm tra (trường hợp 1):", accuracy_test_selected)
print("Báo cáo phân loại (trường hợp 1):")
print(classification_report(y_test_selected, y_test_pred_selected))
[120] ✓ 0.0s
...
Accuracy trên tập huấn luyện (trường hợp 1): 0.9984674329501916
Accuracy trên tập kiểm tra (trường hợp 1): 0.9713993871297242
Báo cáo phân loại (trường hợp 1):
precision    recall    f1-score   support
          0       0.98      0.99      0.98      894
          1       0.89      0.76      0.82      85

   accuracy                           0.97      979
  macro avg       0.93      0.88      0.98      979
weighted avg       0.97      0.97      0.97      979
```

+ Code + Markdown

```
def plot_confusion_matrix(model, y_test, y_pred, case):
    # Tính toán confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Vẽ confusion matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title(f'Confusion Matrix - {type(model).__name__} (Trường hợp {case})')
    plt.show()
[121] ✓ 0.0s
```

plot\_confusion\_matrix(dt\_selected, y\_test\_selected, y\_test\_pred\_selected, 1)

Cell 75 of 131 ⚡ Go Live ⌛ tabnine starter

File Edit Selection View Go Run Terminal Help ← → Search Python 3.11.2 Python

Nhom5.ipynb

```
C:\Users\ngchh\Downloads\ptdfinal> Nhom5.ipynb > M4 6. Thiết lập các mô hình phân loại > M4 6.2. Ma trận tương quan giữa các cột
+ Code + Markdown | ▶ Run All ⚡ Restart ⌛ Clear All Outputs | ⌂ Variables ⌂ Outline ...
weighted avg       0.97      0.97      0.97      979
```

+ Code + Markdown

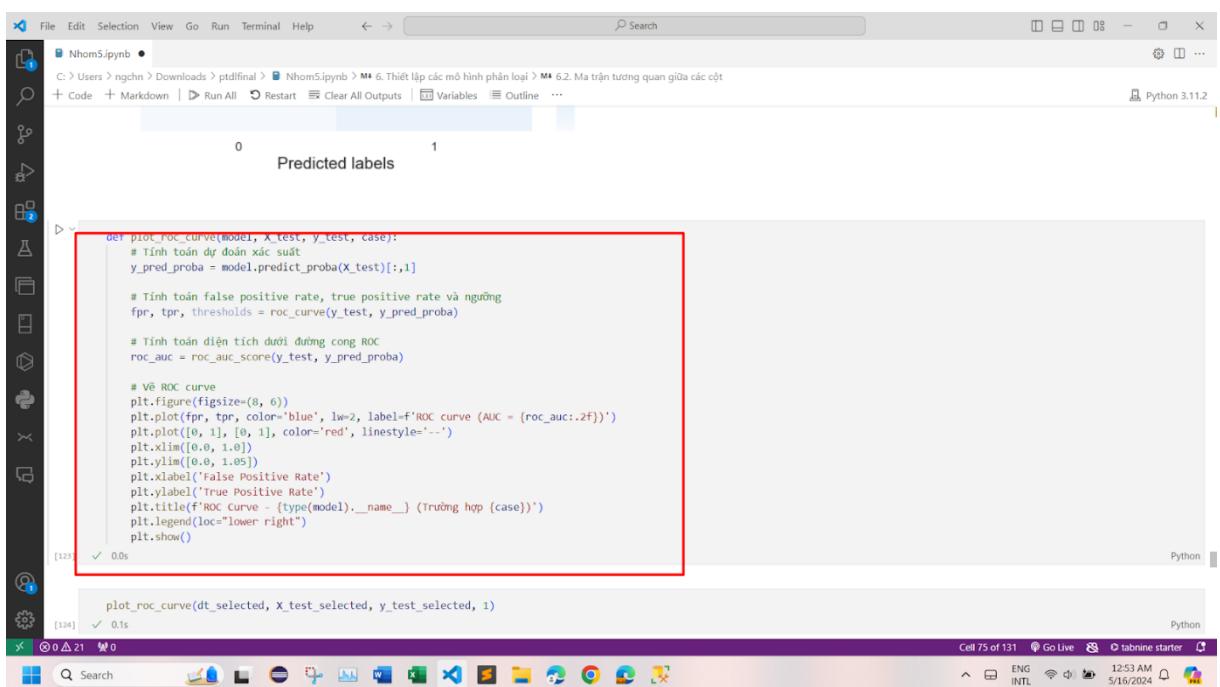
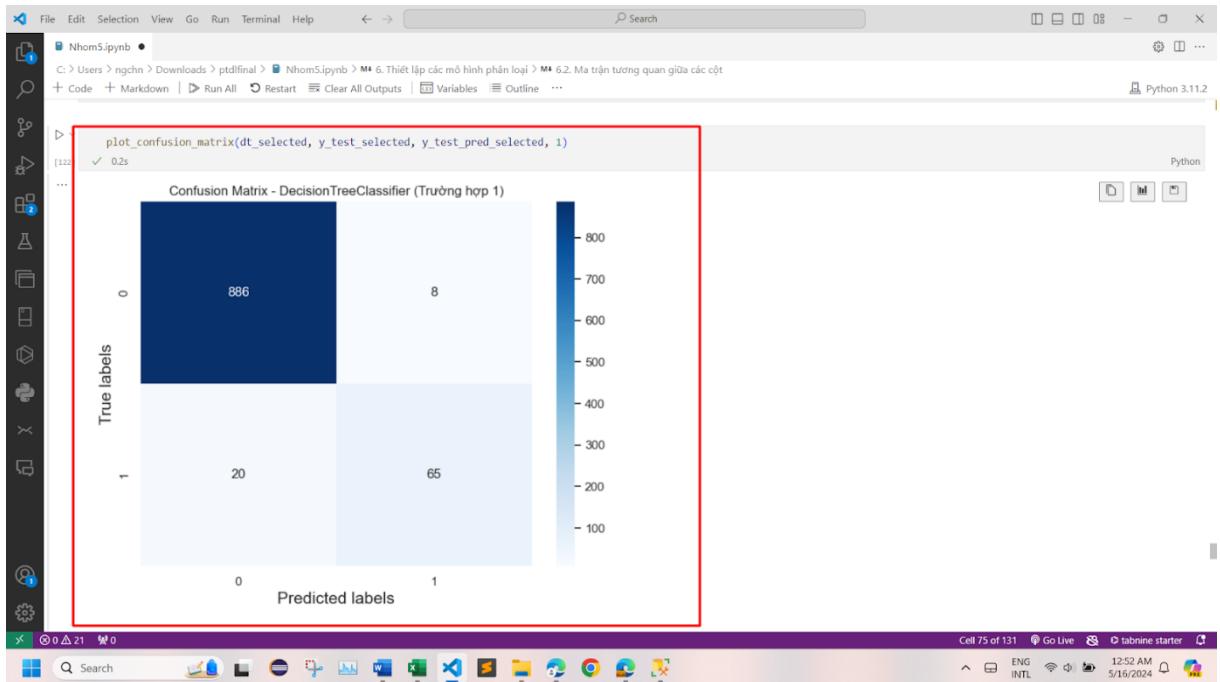
```
def plot_confusion_matrix(model, y_test, y_pred, case):
    # Tính toán confusion matrix
    cm = confusion_matrix(y_test, y_pred)

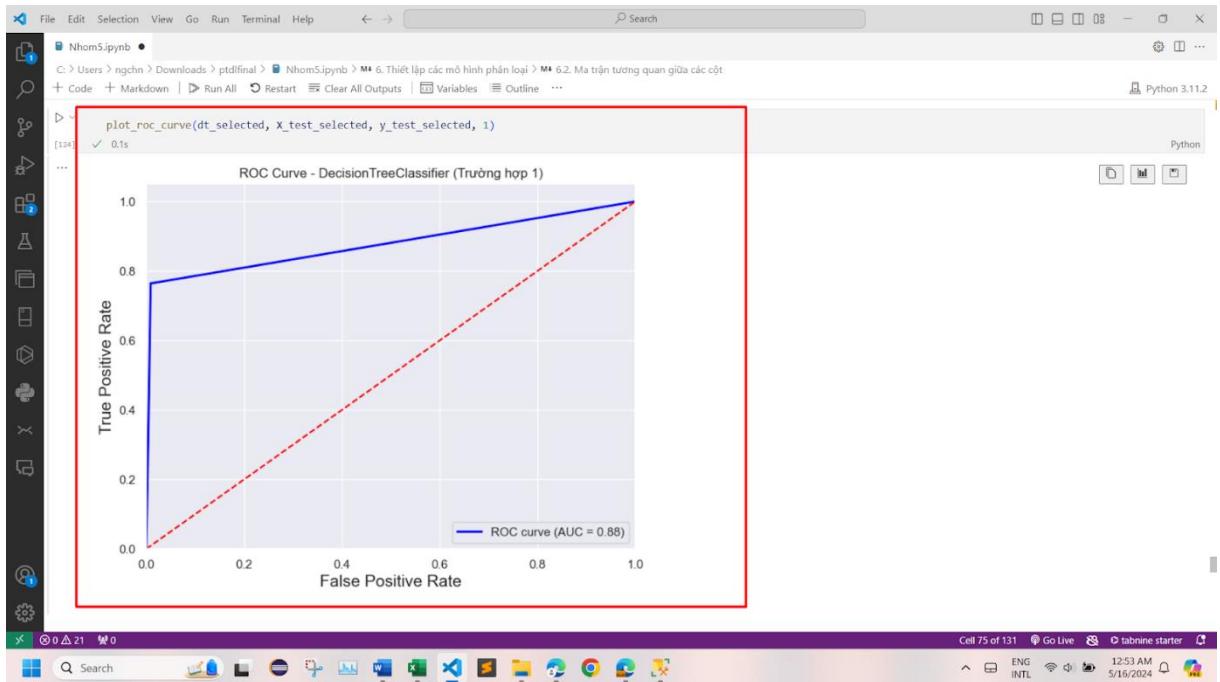
    # Vẽ confusion matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title(f'Confusion Matrix - {type(model).__name__} (Trường hợp {case})')
    plt.show()
[121] ✓ 0.0s
```

D> ...
plot\_confusion\_matrix(dt\_selected, y\_test\_selected, y\_test\_pred\_selected, 1)
[122] ✓ 0.2s

...
Confusion Matrix - DecisionTreeClassifier (Trường hợp 1)

File Edit Selection View Go Run Terminal Help ← → Search Python 3.11.2 Python





#### 2.7.4.2. Trường hợp 2:

```

# Xây dựng mô hình Decision Tree
dt_all = DecisionTreeClassifier(random_state=42)
dt_all.fit(X_train_all, y_train_all)

# Dự đoán trên tập huấn luyện
y_train_pred_all = dt_all.predict(X_train_all)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_all = accuracy_score(y_train_all, y_train_pred_all)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_all)

# Dự đoán trên tập kiểm tra
y_pred_all = dt_all.predict(X_test_all)

# Đánh giá hiệu suất
accuracy_all = accuracy_score(y_test_all, y_pred_all)
print("Accuracy trên tập kiểm tra (trường hợp 2):", accuracy_all)
print("Báo cáo phân loại (trường hợp 2):")
print(classification_report(y_test_all, y_pred_all))

```

Output:

```

Accuracy trên tập huấn luyện (trường hợp 1): 1.0
Accuracy trên tập kiểm tra (trường hợp 2): 0.982635342185904
Báo cáo phân loại (trường hợp 2):
precision    recall   f1-score   support
          0       0.99      0.99      0.99      894
          1       0.99      0.99      0.99      894

```

Nhom5.ipynb

```

accuracy_all = accuracy_score(y_test_all, y_pred_all)
print("Accuracy trên tập huấn luyện (trường hợp 1): 1.0")
print("Accuracy trên tập kiểm tra (trường hợp 2):", accuracy_all)
print("Báo cáo phân loại (trường hợp 2):")
print(classification_report(y_test_all, y_pred_all))

```

...

Accuracy trên tập huấn luyện (trường hợp 1): 1.0  
Accuracy trên tập kiểm tra (trường hợp 2): 0.982635342185904  
Báo cáo phân loại (trường hợp 2):

	precision	recall	f1-score	support
0	0.99	0.99	0.99	894
1	0.89	0.92	0.90	85

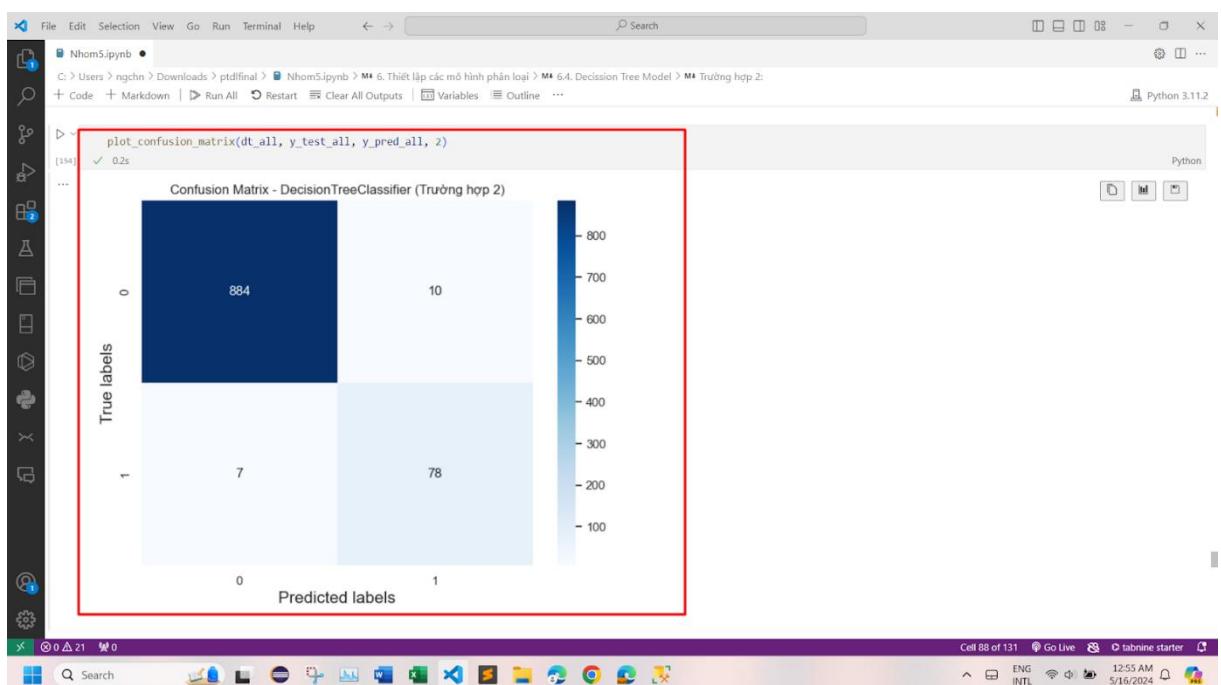
	accuracy	macro avg	weighted avg
0	0.98	0.95	0.98
1	0.99	0.98	0.99

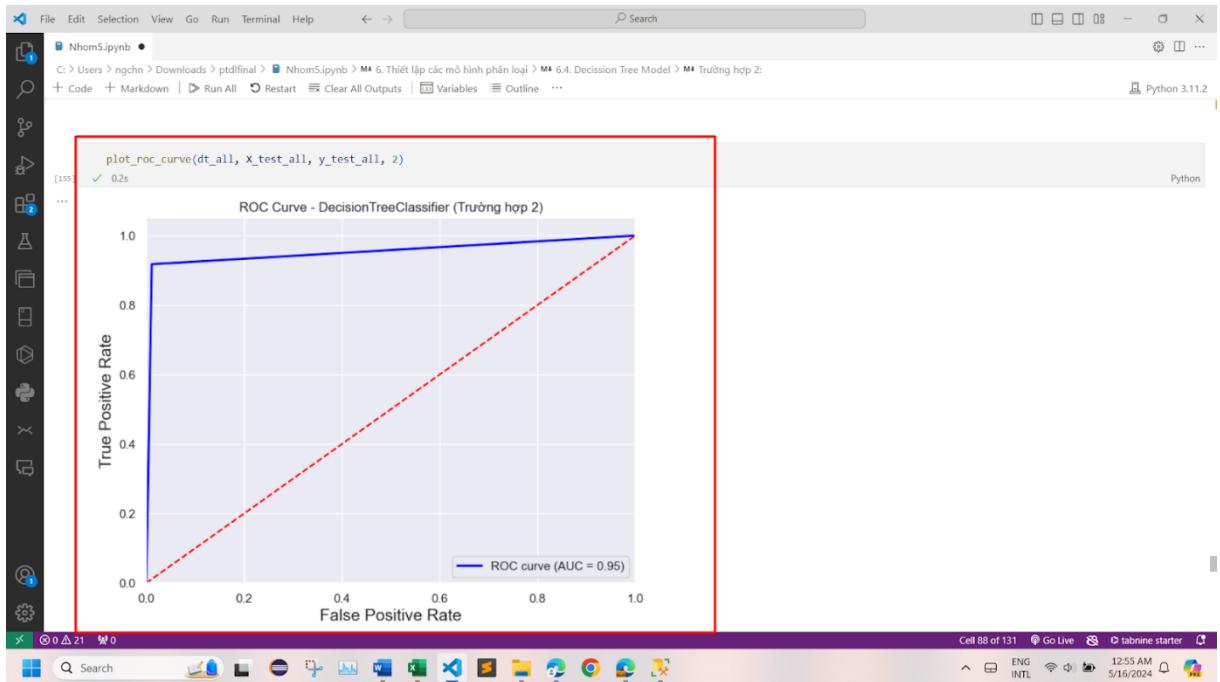
plot\_confusion\_matrix(dt\_all, y\_test\_all, y\_pred\_all, 2)

...

Confusion Matrix - DecisionTreeClassifier (Trường hợp 2)

		True labels
Predicted labels	Confusion Matrix - DecisionTreeClassifier (Trường hợp 2)	
	0	1
0	884	10
1	7	78





Nhận xét: Decission Tree model khi thiết lập với tất cả thuộc tính thì sẽ tốt hơn so với model được tạo bởi các cột có độ tương quan cao

## 2.7.5. XGBClassifier Model

### 2.7.5.1. Trường hợp 1:

```

# Xây dựng mô hình XGBClassifier
xgb_selected = XGBClassifier(random_state=42)
xgb_selected.fit(X_train_selected, y_train_selected)

# Dự đoán trên tập huấn luyện
y_train_pred_selected = xgb_selected.predict(X_train_selected)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_selected = accuracy_score(y_train_selected, y_train_pred_selected)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_selected)

# Dự đoán trên tập kiểm tra
y_pred_selected_xgb = xgb_selected.predict(X_test_selected)

# Đánh giá hiệu suất
accuracy_selected_xgb = accuracy_score(y_test_selected, y_pred_selected_xgb)
print("Accuracy trên tập kiểm tra (trường hợp 1, XGBoost):", accuracy_selected_xgb)
print("Báo cáo phân loại (trường hợp 1, XGBoost):")
print(classification_report(y_test_selected, y_pred_selected_xgb))

```

The code block contains Python code for training an XGBClassifier model and evaluating its accuracy on training and testing datasets. The output shows the accuracy on both training and testing sets, along with a classification report.

Nhom5.ipynb •

C:\> Users > ngchn > Downloads > ptdfinal > Nhom5.ipynb > M4 6. Thiết lập các mô hình phân loại > M4 6.5. XGBClassifier Model

```
+ Code + Markdown | ▶ Run All ⏪ Restart ⏴ Clear All Outputs | ⏴ Variables ⏴ Outline ...
```

```
... print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_selected_xgb)
print("Accuracy trên tập kiểm tra (trường hợp 1, XGBoost):", accuracy_test_xgb)
print("Báo cáo phân loại (trường hợp 1, XGBoost):")
print(classification_report(y_test_selected, y_pred_selected_xgb))
```

[156] ✓ 0.66

```
... Accuracy trên tập huấn luyện (trường hợp 1): 0.9943805874840358
Accuracy trên tập kiểm tra (trường hợp 1, XGBoost): 0.9765066394279878
Báo cáo phân loại (trường hợp 1, XGBoost):
precision    recall   f1-score   support
```

	precision	recall	f1-score	support
0	0.98	0.99	0.99	894
1	0.92	0.89	0.86	85

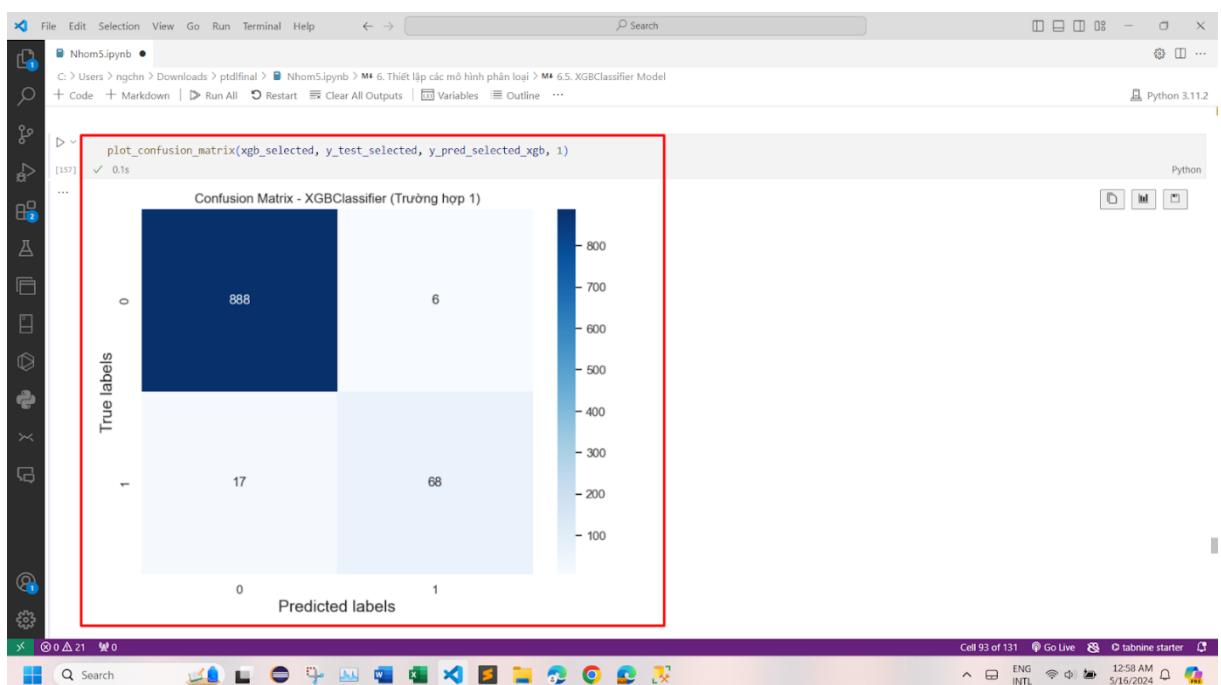
```
accuracy           0.98
macro avg       0.95
weighted avg   0.98
```

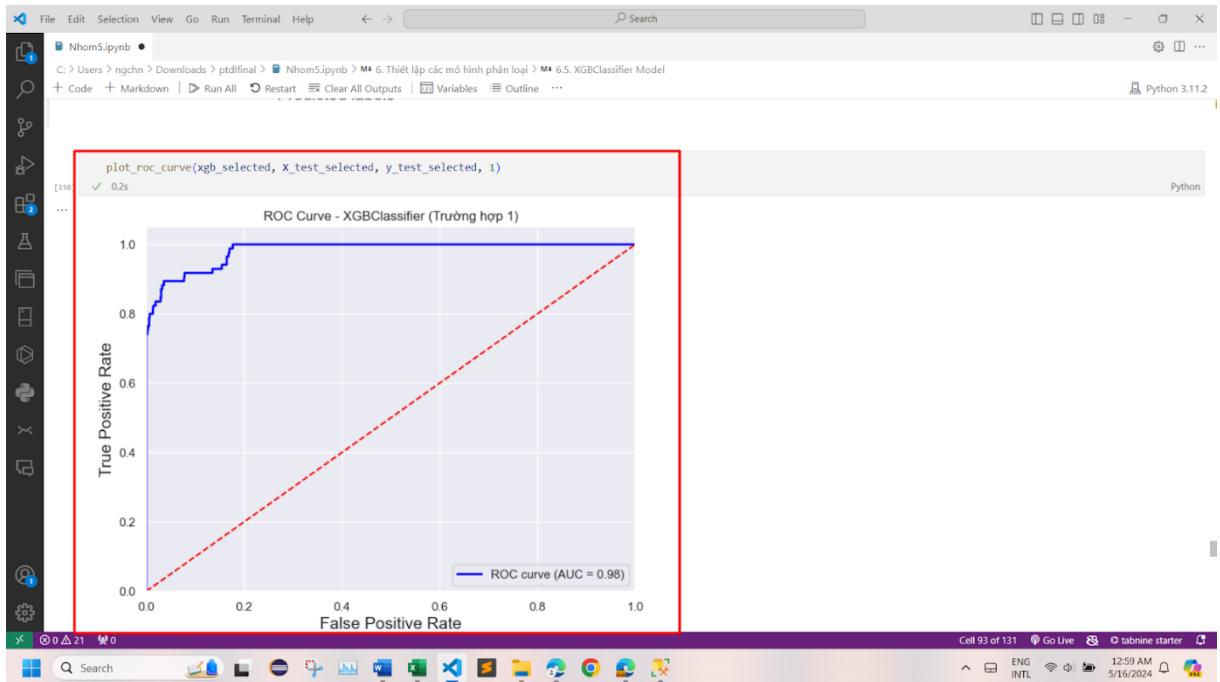
plot\_confusion\_matrix(xgb\_selected, y\_test\_selected, y\_pred\_selected\_xgb, 1)

[157] ✓ 0.1s

... Confusion Matrix - XGBClassifier (Trường hợp 1)

		Predicted labels	
		0	1
True labels	0	888	6
	1	17	68





### 2.7.5.2. Trường hợp 2:

```
# Xây dựng mô hình XGBClassifier
xgb_all = XGBClassifier(random_state=42)
xgb_all.fit(X_train_all, y_train_all)

# Dự đoán trên tập huấn luyện
y_train_pred_all = xgb_all.predict(X_train_all)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_all = accuracy_score(y_train_all, y_train_pred_all)

# In accuracy trên tập huấn luyện (trường hợp 1):
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_all)

# Dự đoán trên tập kiểm tra
y_pred_all_xgb = xgb_all.predict(X_test_all)

# Đánh giá hiệu suất
accuracy_all_xgb = accuracy_score(y_test_all, y_pred_all_xgb)
print("Accuracy trên tập kiểm tra (trường hợp 2, XGBoost):", accuracy_all_xgb)
print("Báo cáo phân loại (trường hợp 2, XGBoost):")
print(classification_report(y_test_all, y_pred_all_xgb))

```

...

	precision	recall	f1-score	support
0	1.00	0.99	0.99	894
1	0.93	0.95	0.94	85

Nhom5.ipynb

```
# Đánh giá hiệu suất
accuracy_all_xgb = accuracy_score(y_test_all, y_pred_all_xgb)
print("Accuracy trên tập huấn luyện (trường hợp 1): 1.0")
print("Accuracy trên tập kiểm tra (trường hợp 2, XGBoost):", accuracy_all_xgb)
print("Báo cáo phân loại (trường hợp 2, XGBoost):")
print(classification_report(y_test_all, y_pred_all_xgb))
[159] ✓ 0.1s
```

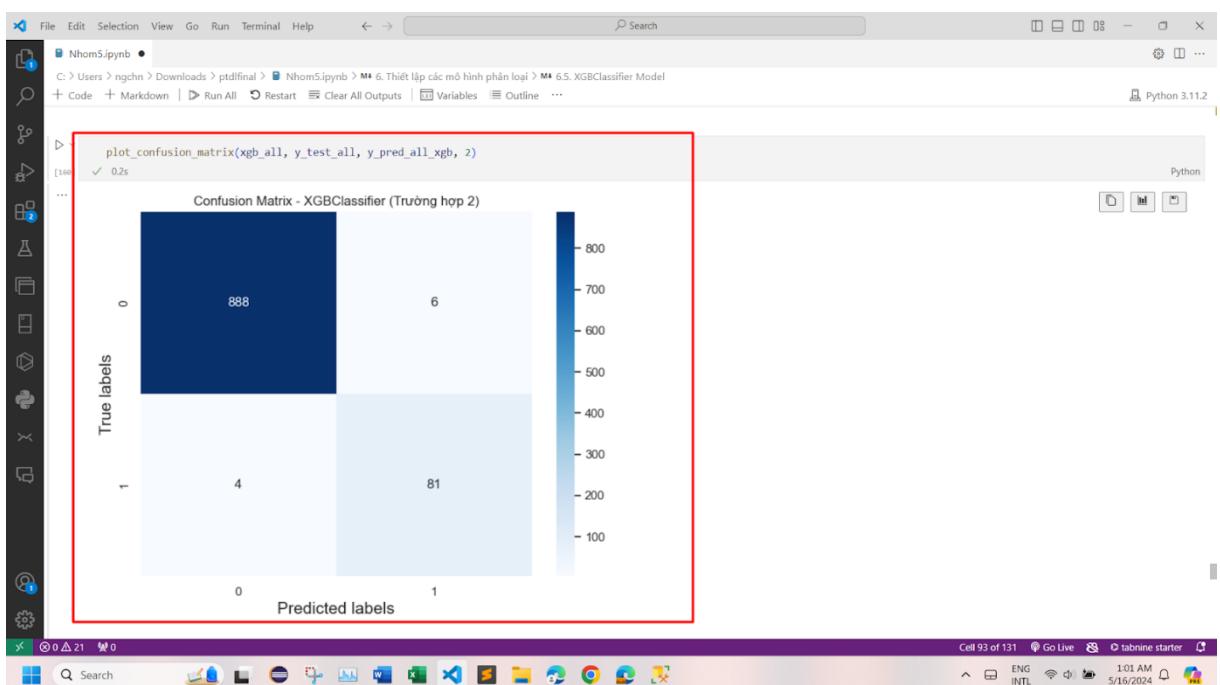
... Accuracy trên tập huấn luyện (trường hợp 1): 1.0  
Accuracy trên tập kiểm tra (trường hợp 2, XGBoost): 0.9897854954034729  
Báo cáo phân loại (trường hợp 2, XGBoost):

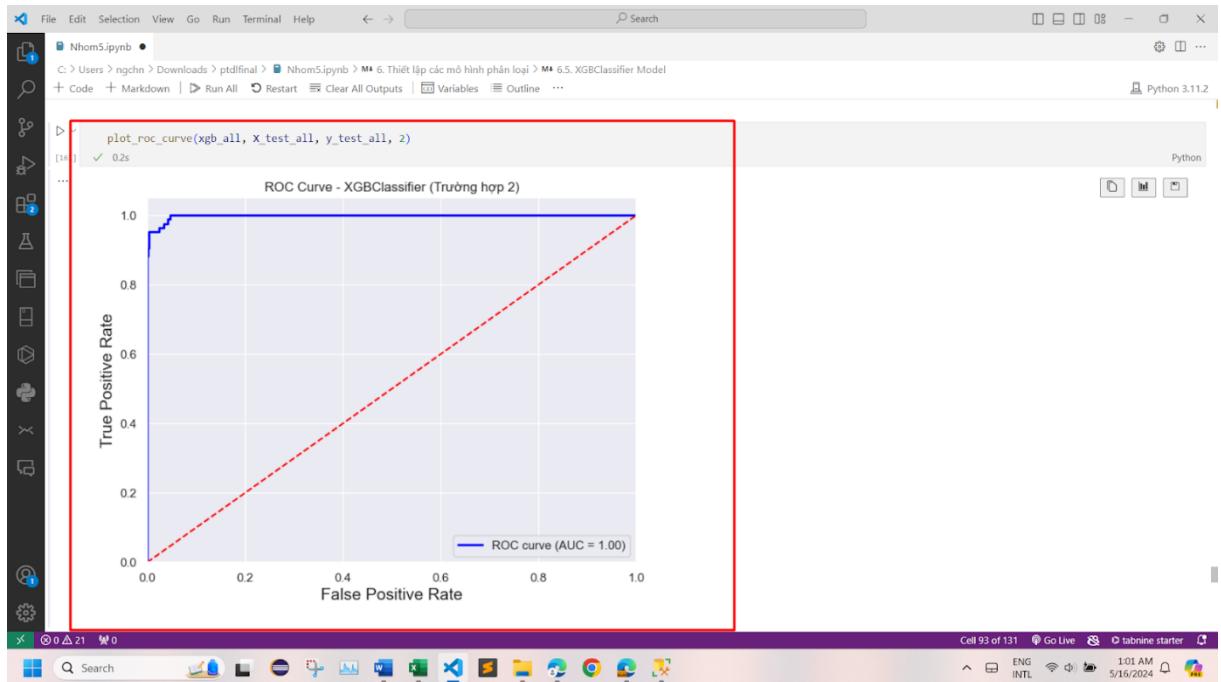
	precision	recall	f1-score	support
0	1.00	0.99	0.99	894
1	0.93	0.95	0.94	85
accuracy			0.99	979
macro avg	0.96	0.97	0.97	979
weighted avg	0.99	0.99	0.99	979

```
plot_confusion_matrix(xgb_all, y_test_all, y_pred_all_xgb, 2)
```

[160] ✓ 0.2s

... Confusion Matrix - XGBClassifier (Trường hợp 2)





Nhận xét: Model với tất cả thuộc tính có AUC cao hơn

## 2.7.6. KNeighborsClassifier Model

### 2.7.6.1. Trường hợp 1:

```

# Xây dựng mô hình KNeighborsClassifier
knn_selected = KNeighborsClassifier()
knn_selected.fit(X_train_selected, y_train_selected)

# Dự đoán trên tập huấn luyện
y_train_pred_selected = knn_selected.predict(X_train_selected)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_selected = accuracy_score(y_train_selected, y_train_pred_selected)

# In accuracy trên tập huấn luyện (trường hợp 1):
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_selected)

# Dự đoán trên tập kiểm tra
y_pred_selected_knn = knn_selected.predict(X_test_selected)

# Đánh giá hiệu suất
accuracy_selected_knn = accuracy_score(y_test_selected, y_pred_selected_knn)
print("Accuracy trên tập kiểm tra (trường hợp 1, Kneighbors):", accuracy_selected_knn)
print("Báo cáo phân loại (trường hợp 1, Kneighbors):")
print(classification_report(y_test_selected, y_pred_selected_knn))

```

Nhom5.ipynb

```
# Đánh giá hiệu suất
accuracy_selected_knn = accuracy_score(y_test_selected, y_pred_selected_knn)
print("Accuracy trên tập kiểm tra (trường hợp 1, KNeighbors):", accuracy_selected_knn)
print("Báo cáo phân loại (trường hợp 1, KNeighbors):")
print(classification_report(y_test_selected, y_pred_selected_knn))

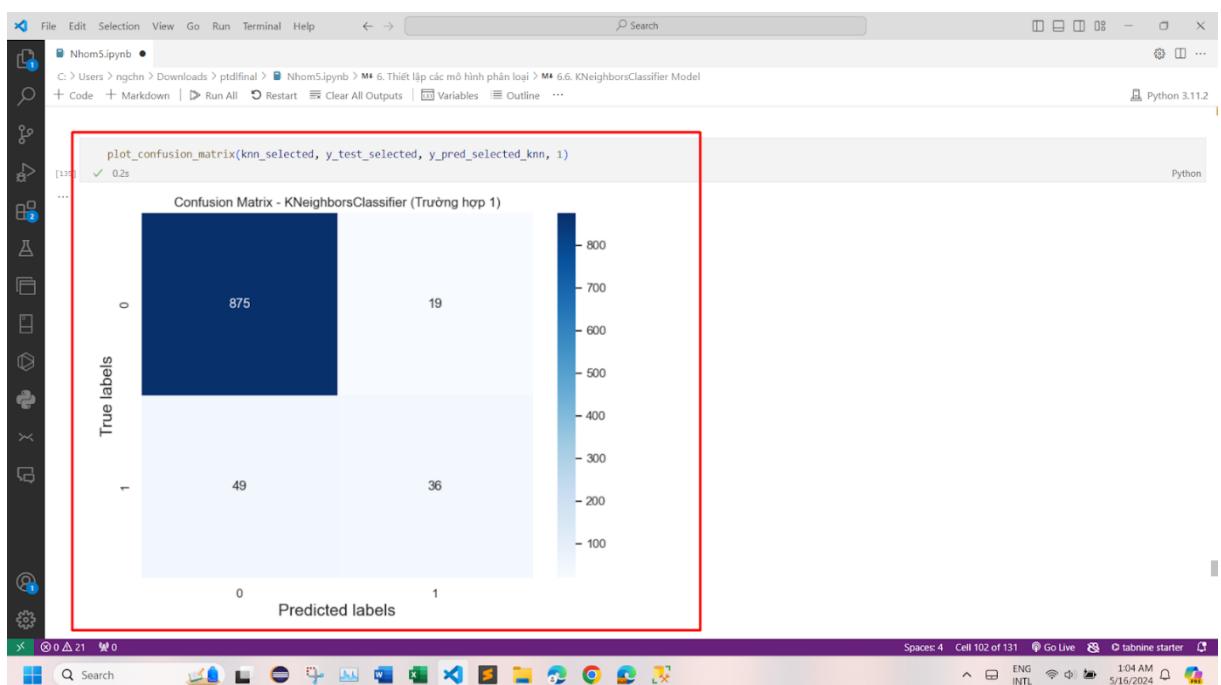
[134] ✓ 0.1s
Accuracy trên tập huấn luyện (trường hợp 1): 0.9491698595146871
Accuracy trên tập kiểm tra (trường hợp 1, KNeighbors): 0.93805413687436159
Báo cáo phân loại (trường hợp 1, KNeighbors):
precision recall f1-score support

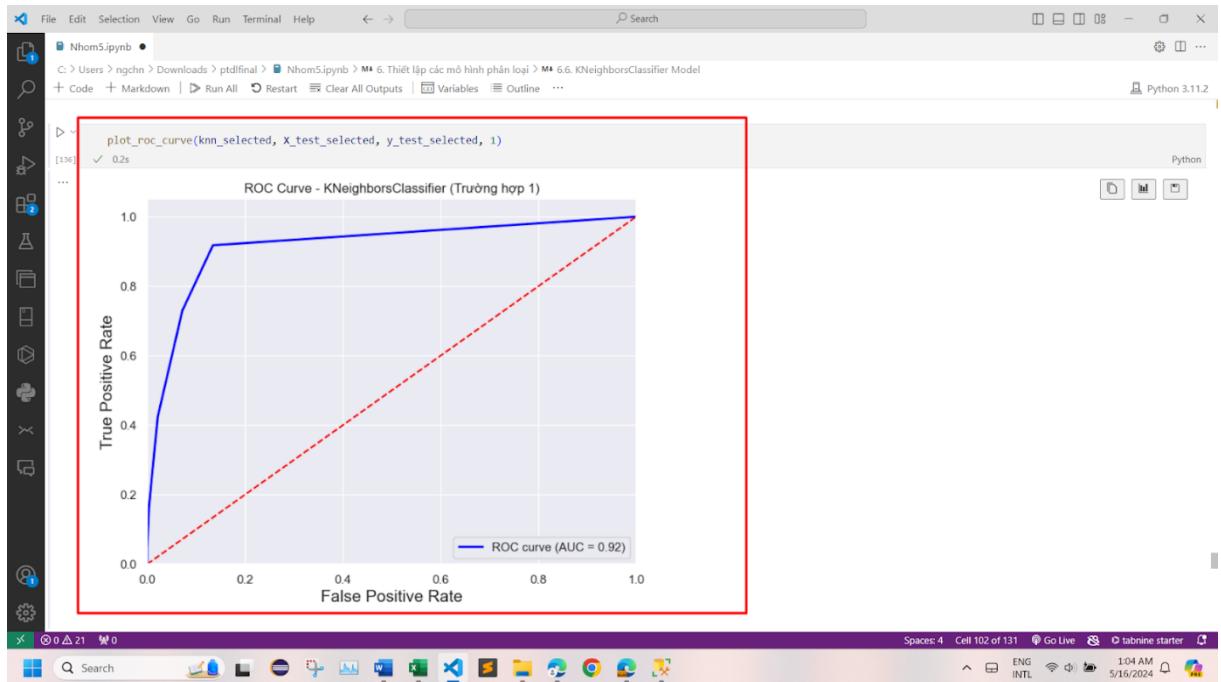
          0      0.95     0.98    0.96    894
          1      0.65     0.42    0.51     85

   accuracy
macro avg  0.80     0.70    0.79    979
weighted avg 0.92     0.93    0.92    979

plot_confusion_matrix(knn_selected, y_test_selected, y_pred_selected_knn, 1)

[135] ✓ 0.2s
...
Confusion Matrix - KNeighborsClassifier (Trường hợp 1)
```





### 2.7.6.2. Trường hợp 2:

```
# Xây dựng mô hình KNeighborsClassifier
knn_all = KNeighborsClassifier()
knn_all.fit(X_train_all, y_train_all)

# Dự đoán trên tập huấn luyện
y_train_pred_all = knn_all.predict(X_train_all)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_all = accuracy_score(y_train_all, y_train_pred_all)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_all)

# Dự đoán trên tập kiểm tra
y_pred_all_knn = knn_all.predict(X_test_all)

# Đánh giá hiệu suất
accuracy_all_knn = accuracy_score(y_test_all, y_pred_all_knn)
print("Accuracy trên tập kiểm tra (trường hợp 2, KNeighbors):", accuracy_all_knn)
print("Báo cáo phân loại (trường hợp 2, KNeighbors):")
print(classification_report(y_test_all, y_pred_all_knn))
```

Nhom5.ipynb

```
C:\> Users > ngchn > Downloads > ptdfinal > Nhom5.ipynb > M4. Thiết lập các mô hình phân loại > M4.6. KNeighborsClassifier Model
```

```
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...
```

```
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_all_knn)
print("Báo cáo phân loại (trường hợp 2, Kneighbors):")
print(classification_report(y_test_all, y_pred_all_knn))
```

```
[137] ✓ 0.1s
```

```
Accuracy trên tập huấn luyện (trường hợp 1): 0.9320561941251596
Accuracy trên tập kiểm tra (trường hợp 2, Kneighbors): 0.9172625127681308
Báo cáo phân loại (trường hợp 2, Kneighbors):
precision    recall   f1-score   support
```

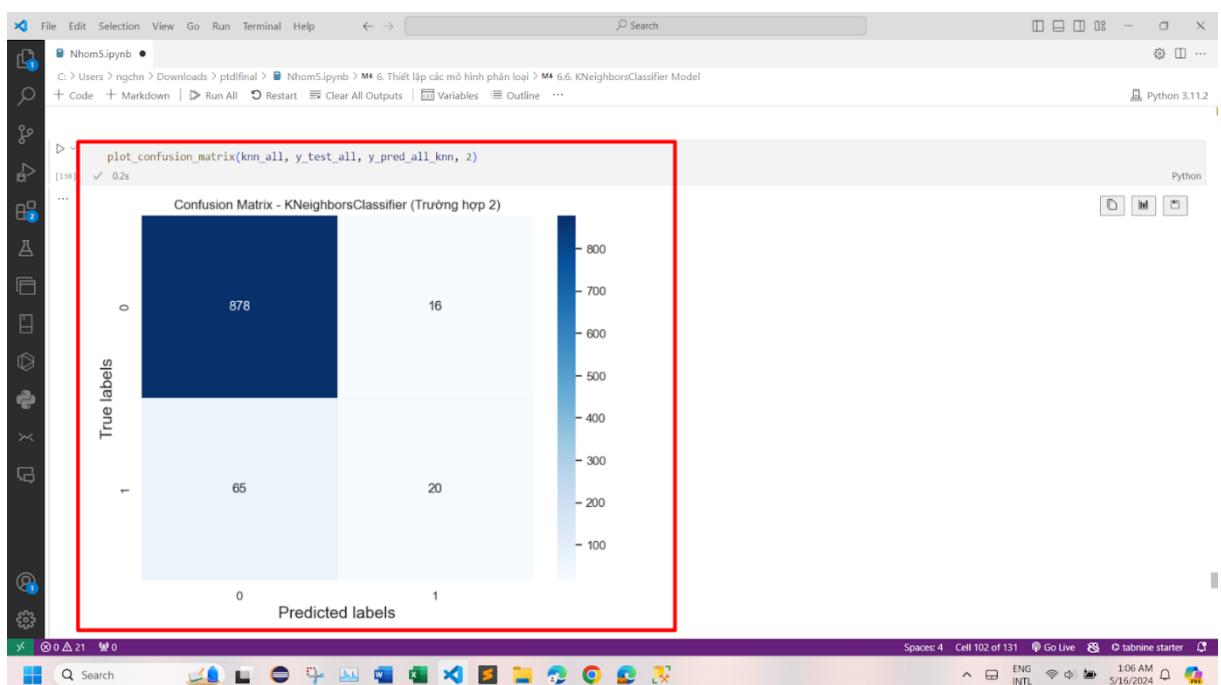
	0	0.93	0.98	0.96	894
1	0.56	0.24	0.33	85	
accuracy				0.92	979
macro avg	0.74	0.61	0.64	0.979	979
weighted avg	0.90	0.92	0.90	0.979	979

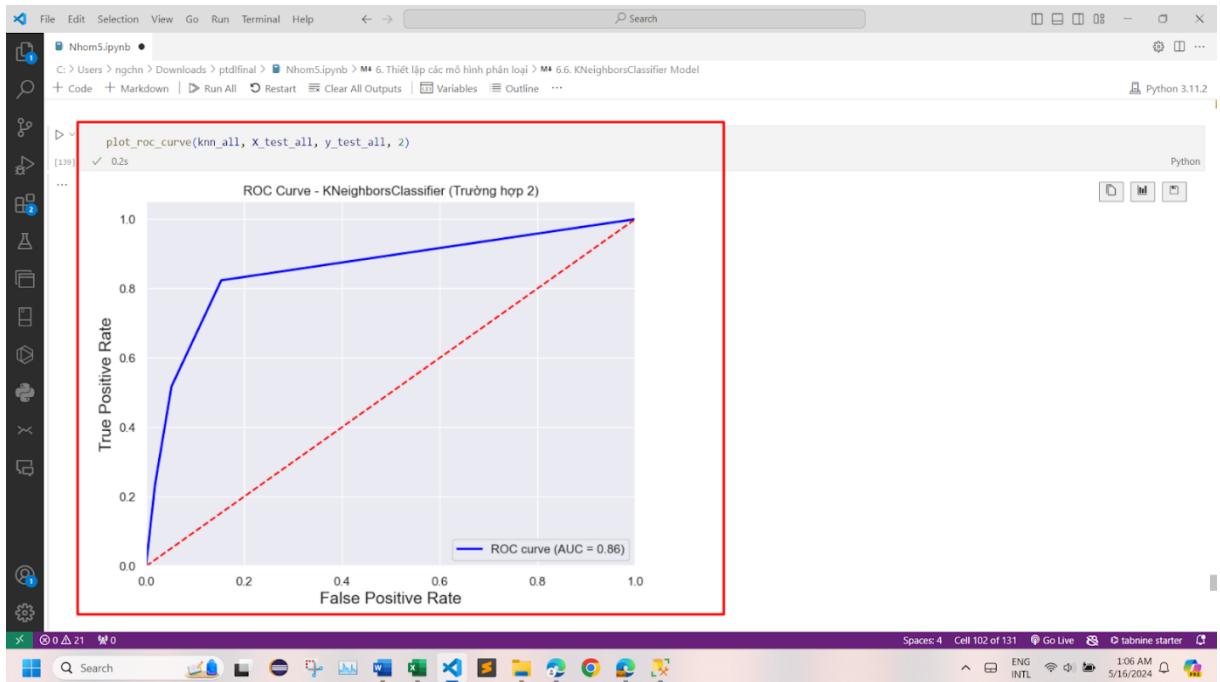
```
plot_confusion_matrix(knn_all, y_test_all, y_pred_all_knn, 2)
```

```
[138] ✓ 0.2s
```

```
Confusion Matrix - KNeighborsClassifier (Trường hợp 2)
```

		0	1
0	878	16	
1	65	20	





Nhận xét: Với KNeighbors Classifier Model thì sử dụng các cột tương quan cao thì cho ra AUC cao hơn

## 2.7.7. LogisticRegression Model

### 2.7.7.1. Trường hợp 1:

```
# Xây dựng mô hình Logistic Regression
logreg_selected = LogisticRegression()
logreg_selected.fit(X_train_selected, y_train_selected)

# Dự đoán trên tập huấn luyện
y_train_pred_selected = logreg_selected.predict(X_train_selected)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_selected = accuracy_score(y_train_selected, y_train_pred_selected)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_selected)

# Dự đoán trên tập kiểm tra
y_pred_selected_logreg = logreg_selected.predict(X_test_selected)

# Đánh giá hiệu suất
accuracy_selected_logreg = accuracy_score(y_test_selected, y_pred_selected_logreg)
print("Accuracy trên tập kiểm tra (trường hợp 1, Logistic Regression):", accuracy_selected_logreg)
print("Báo cáo phân loại (trường hợp 1, logistic Regression):")
print(classification_report(y_test_selected, y_pred_selected_logreg))
```

Nhom5.ipynb

```

C: > Users > ngchn > Downloads > ptdfinal > Nhom5.ipynb > M4 6. Thiết lập các mô hình phân loại > M4 6.7. LogisticRegression Model
+ Code + Markdown | ▶ Run All ⚡ Restart ⚡ Clear All Outputs | Variables Outline ...
D> y_train_pred_selected = logreg_selected.predict(X_train_selected)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_selected = accuracy_score(y_train_selected, y_train_pred_selected)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_selected)

# Dự đoán trên tập kiểm tra
y_pred_selected_logreg = logreg_selected.predict(X_test_selected)

# Đánh giá hiệu suất
accuracy_selected_logreg = accuracy_score(y_test_selected, y_pred_selected_logreg)
print("Accuracy trên tập kiểm tra (trường hợp 1, Logistic Regression):", accuracy_selected_logreg)
print("Báo cáo phân loại (trường hợp 1, Logistic Regression):")
print(classification_report(y_test_selected, y_pred_selected_logreg))

```

[140] ✓ 0.0s

```

...
Accuracy trên tập huấn luyện (trường hợp 1): 0.9499361430395913
Accuracy trên tập kiểm tra (trường hợp 1, Logistic Regression): 0.9560776302349336
Báo cáo phân loại (trường hợp 1, Logistic Regression):
precision recall f1-score support
0 0.96 0.99 0.98 894
1 0.85 0.60 0.70 85

accuracy 0.96 0.97 0.97
macro avg 0.91 0.79 0.84 979
weighted avg 0.95 0.96 0.95 979

```

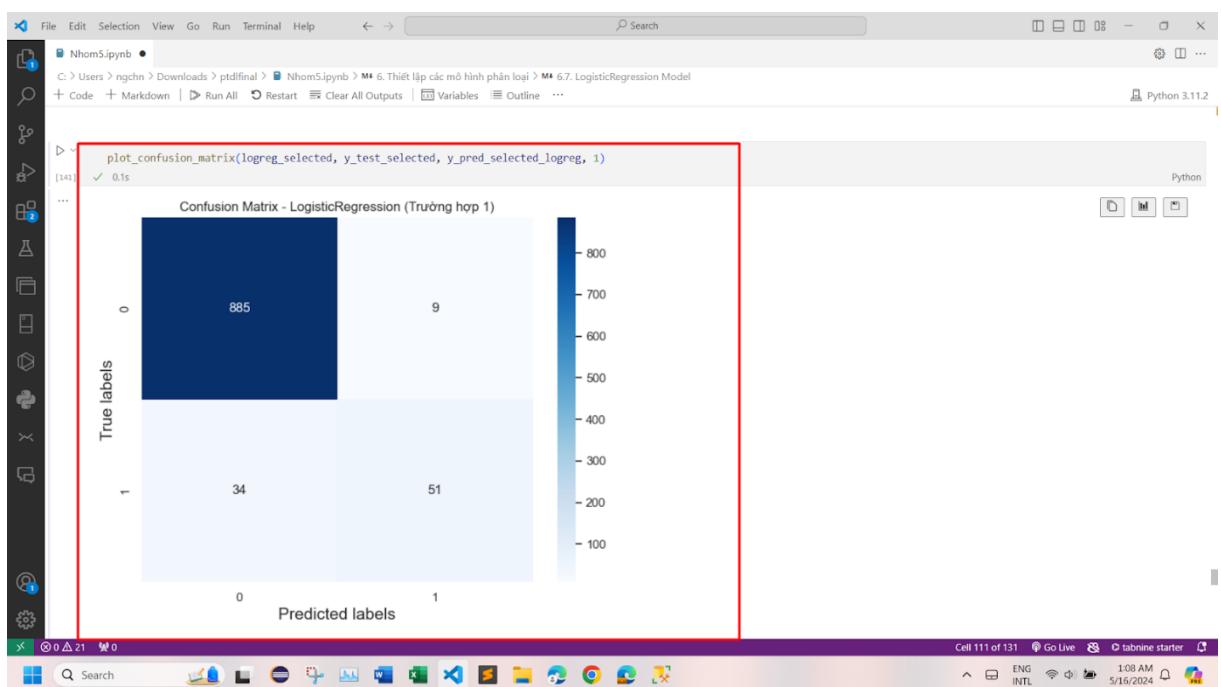
[141] ✓ 0.1s

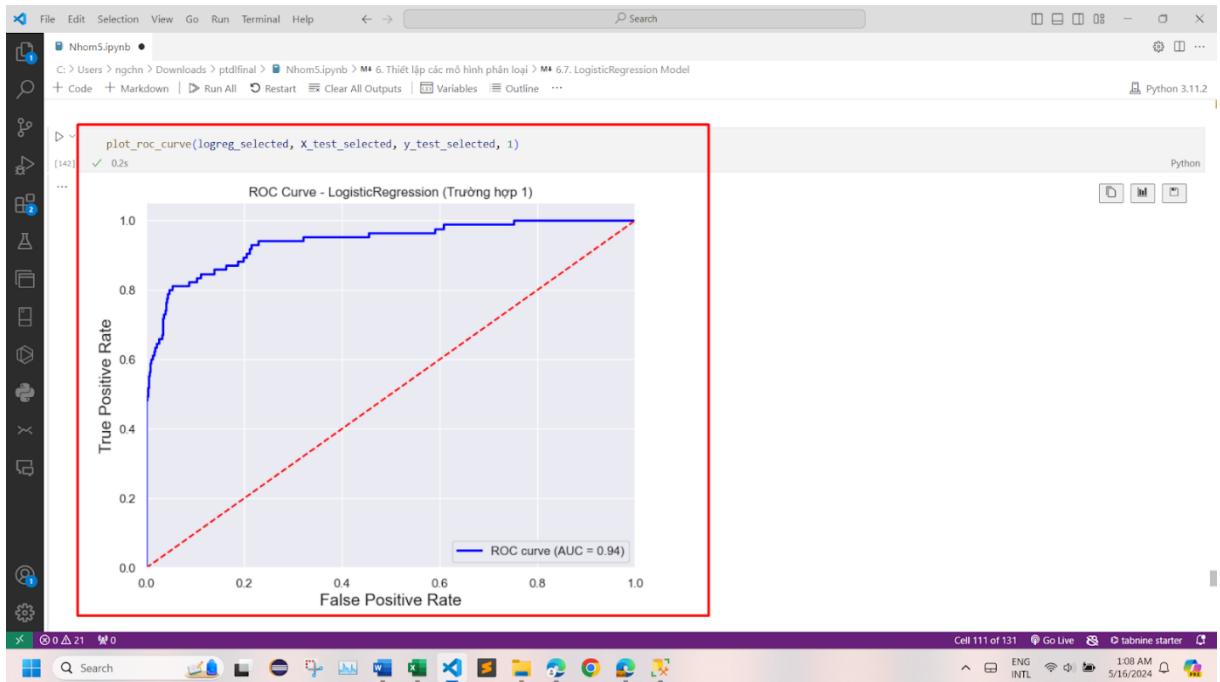
```

plot_confusion_matrix(logreg_selected, y_test_selected, y_pred_selected_logreg, 1)

```

Cell 111 of 131 Go Live tabnine starter Python





### 2.7.7.2. Trường hợp 2:

```
# Xây dựng mô hình logistic Regression
logreg_all = LogisticRegression()
logreg_all.fit(X_train_all, y_train_all)

# Dự đoán trên tập huấn luyện
y_train_pred_all = logreg_all.predict(X_train_all)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_all = accuracy_score(y_train_all, y_train_pred_all)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_all)

# Dự đoán trên tập kiểm tra
y_pred_all_logreg = logreg_all.predict(X_test_all)

# Đánh giá hiệu suất
accuracy_all_logreg = accuracy_score(y_test_all, y_pred_all_logreg)
print("Accuracy trên tập kiểm tra (trường hợp 2, Logistic Regression):", accuracy_all_logreg)
print("Báo cáo phân loại (trường hợp 2, Logistic Regression):")
print(classification_report(y_test_all, y_pred_all_logreg))

[14] ✓ 0.0s
```

Accuracy trên tập huấn luyện (trường hợp 1): 0.9425287356321839  
 Accuracy trên tập kiểm tra (trường hợp 2, Logistic Regression): 0.9448416751787538  
 Báo cáo phân loại (trường hợp 2, Logistic Regression):

	precision	recall	f1-score	support
0	0.96	0.98	0.97	894
1	0.74	0.56	0.64	85

Nhom5.ipynb •

C:\Users\ngchn> Downloads > ptdfinal > Nhom5.ipynb > M4 6. Thiết lập các mô hình phân loại > M4 6.7. LogisticRegression Model

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ...

[143] ✓ 0.0s

```
# Đánh giá hiệu suất
accuracy_all_logreg = accuracy_score(y_test_all, y_pred_all_logreg)
print("Accuracy trên tập kiểm tra (trường hợp 2, Logistic Regression):", accuracy_all_logreg)
print("Báo cáo phân loại (trường hợp 2, Logistic Regression):")
print(classification_report(y_test_all, y_pred_all_logreg))
```

Accuracy trên tập huấn luyện (trường hợp 1): 0.9425287356321839  
Accuracy trên tập kiểm tra (trường hợp 2, logistic Regression): 0.9448416751787538  
Báo cáo phân loại (trường hợp 2, Logistic Regression):

	precision	recall	f1-score	support
0	0.96	0.98	0.97	894
1	0.74	0.56	0.64	85
accuracy			0.94	979
macro avg	0.85	0.77	0.81	979
weighted avg	0.94	0.94	0.94	979

e:\Users\ngchn\AppData\Local\Programs\Python\Python311\lib\site-packages\sklearn\linear\_model\logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:  
<https://scikit-learn.org/stable/modules/preprocessing.html>  
Please also refer to the documentation for alternative solver options:  
[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)  
n\_iter\_i = \_check\_optimize\_result(

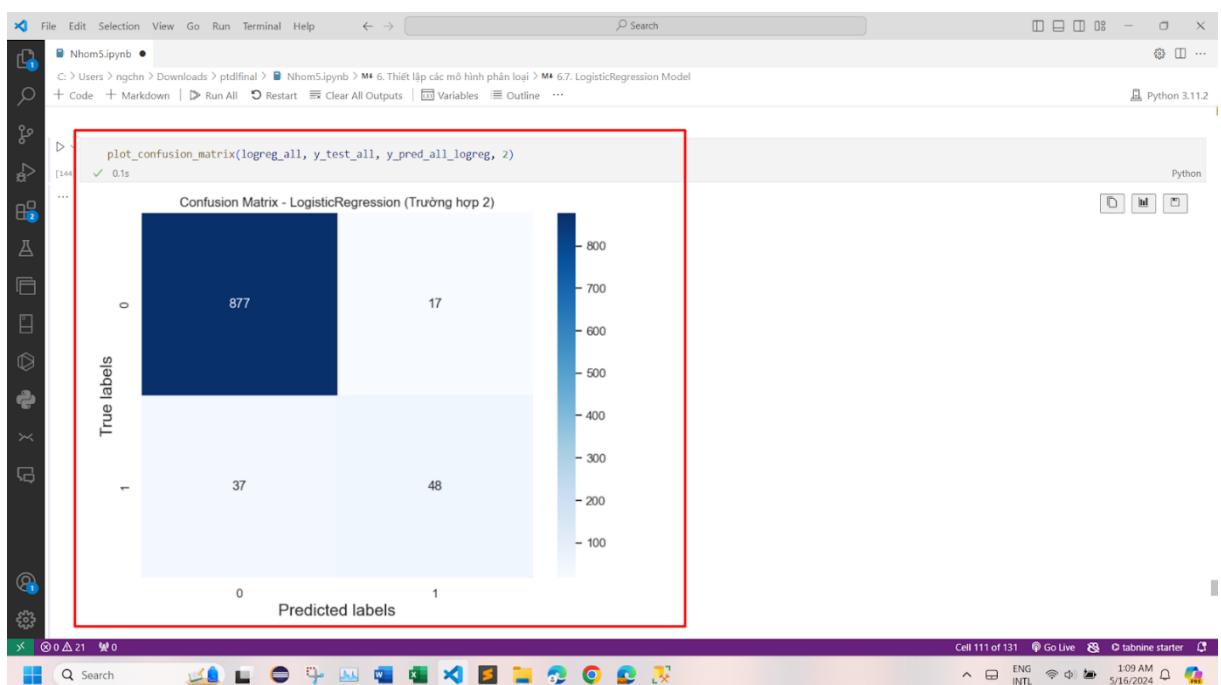
[144] ✓ 0.1s

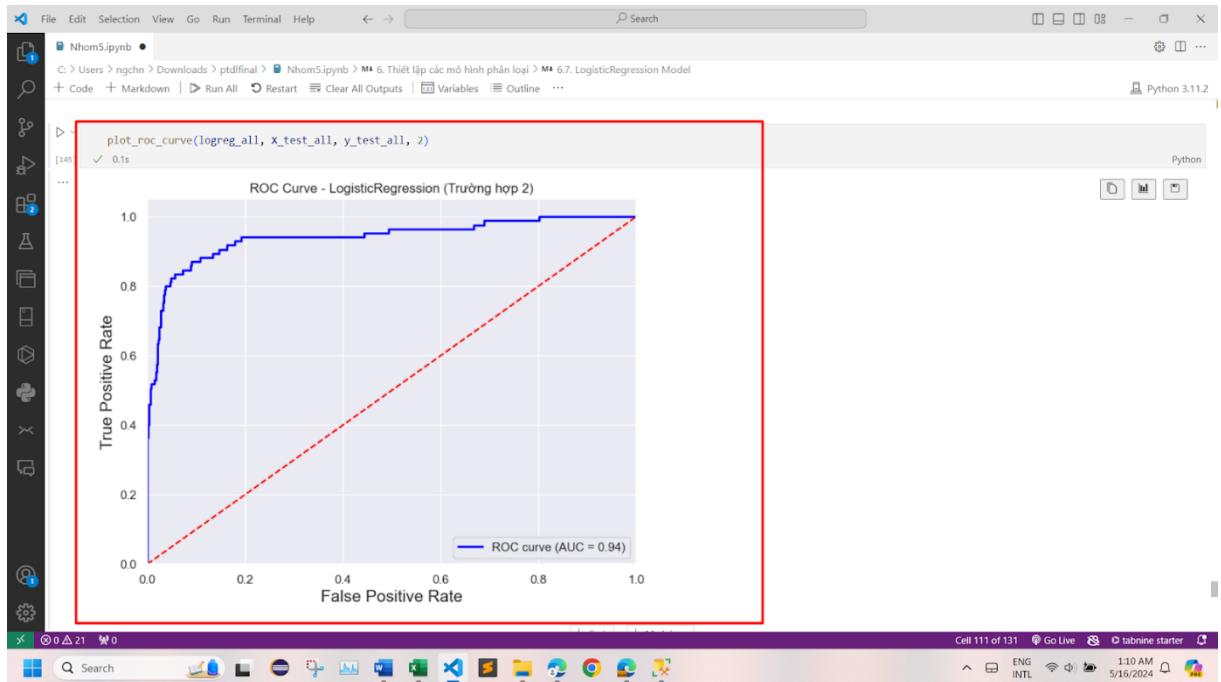
```
plot_confusion_matrix(logreg_all, y_test_all, y_pred_all_logreg, 2)
```

... Confusion Matrix - LogisticRegression (Trường hợp 2)

Cell 111 of 131 Go Live tabnine starter

0 21 0 0 109 AM 5/16/2024 ENG INTL





Nhận xét: Logistic Regression Model thì cho ra AUC bằng nhau

## 2.7.8. RandomForest Model

### 2.7.8.1. Trường hợp 1:

```

# Xây dựng mô hình RandomForest
rf_selected = RandomForestClassifier(random_state=42)
rf_selected.fit(X_train_selected, y_train_selected)

# Dự đoán trên tập huấn luyện
y_train_pred_selected = rf_selected.predict(X_train_selected)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_selected = accuracy_score(y_train_selected, y_train_pred_selected)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_selected)

# Dự đoán trên tập kiểm tra
y_pred_selected_rf = rf_selected.predict(X_test_selected)

# Đánh giá hiệu suất
accuracy_selected_rf = accuracy_score(y_test_selected, y_pred_selected_rf)
print("Accuracy trên tập kiểm tra (trường hợp 1, RandomForest):", accuracy_selected_rf)
print("Báo cáo phân loại (trường hợp 1, RandomForest):")
print(classification_report(y_test_selected, y_pred_selected_rf))

```

Nhom5.ipynb

```
C:\Users\ngchn>Downloads>ptdfinal> Nhom5.ipynb>...
+ Code + Markdown | ▶ Run All ⚡ Restart ⚡ Clear All Outputs | Variables Outline ...
D> v print("Độ chính xác trên tập huấn luyện (trường hợp 1): 0.9984674329501916")
print("Độ chính xác trên tập kiểm tra (trường hợp 1, RandomForest): 0.975485188968335")
print("Báo cáo phân loại (trường hợp 1, RandomForest):")
precision recall f1-score support
...
0 0.98 0.99 0.99 894
1 0.92 0.79 0.85 85
...
accuracy 0.95 0.89 0.92 979
macro avg 0.97 0.98 0.97 979
weighted avg 0.97 0.98 0.97 979
```

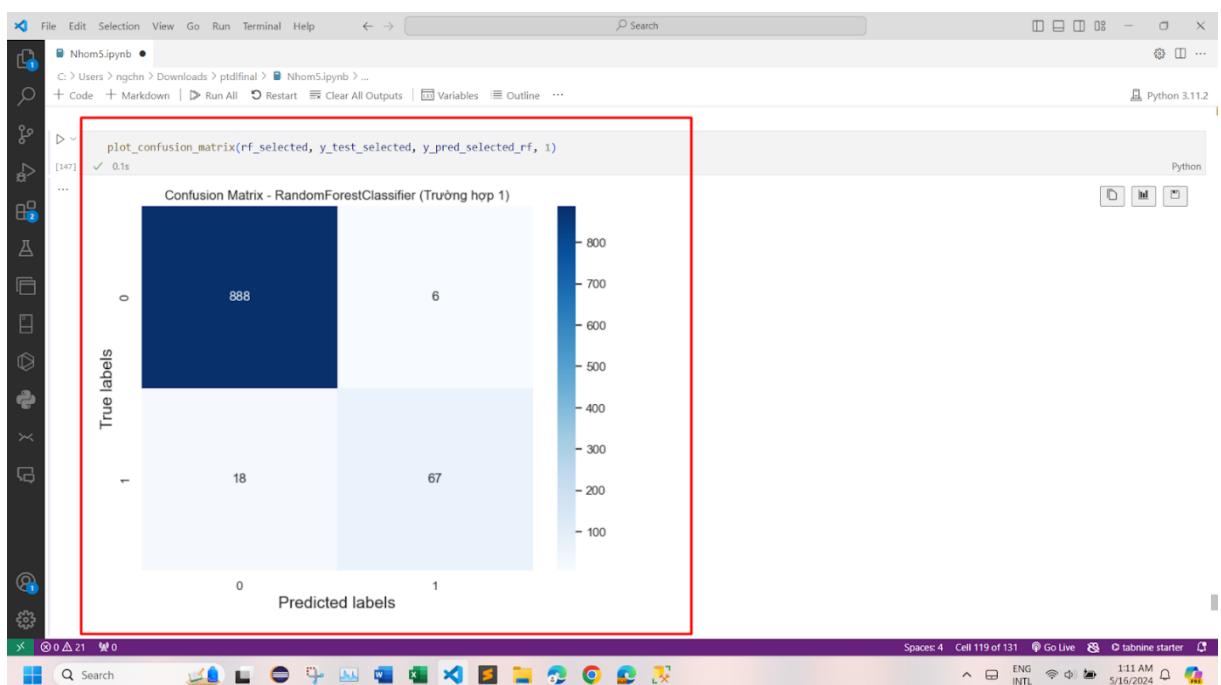
plot\_confusion\_matrix(rf\_selected, y\_test\_selected, y\_pred\_selected\_rf, 1)

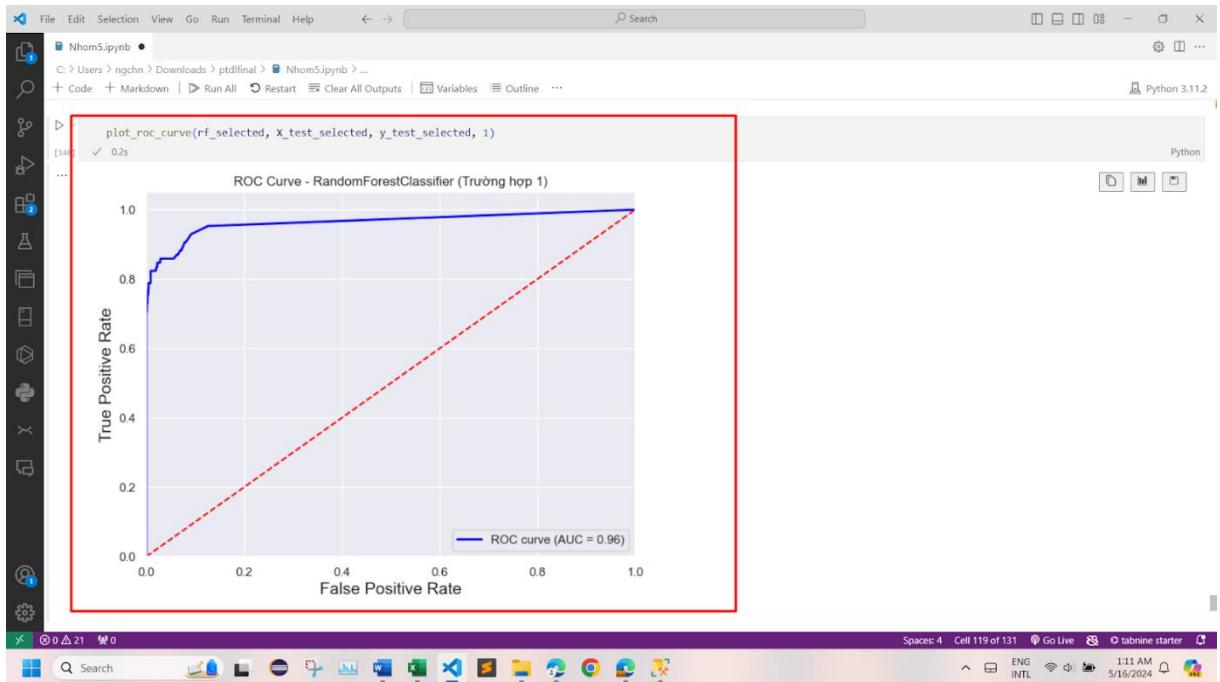
... Confusion Matrix - RandomForestClassifier (Trường hợp 1)

		0	1
0	888	6	
1	18	67	
True labels		0	1

Spaces: 4 Cell 119 of 131 Go Live tabnine starter

11:11 AM 5/16/2024





### 2.7.8.2. Trường hợp 2:

```
# Xây dựng mô hình RandomForest
rf_all = RandomForestClassifier(random_state=42)
rf_all.fit(X_train_all, y_train_all)

# Dự đoán trên tập huấn luyện
y_train_pred_all = rf_all.predict(X_train_all)

# Đánh giá hiệu suất trên tập huấn luyện
accuracy_train_all = accuracy_score(y_train_all, y_train_pred_all)

# In accuracy trên tập huấn luyện
print("Accuracy trên tập huấn luyện (trường hợp 1):", accuracy_train_all)

# Dự đoán trên tập kiểm tra
y_pred_all_rf = rf_all.predict(X_test_all)

# Đánh giá hiệu suất
accuracy_all_rf = accuracy_score(y_test_all, y_pred_all_rf)
print("Accuracy trên tập kiểm tra (trường hợp 2, RandomForest):", accuracy_all_rf)
print("Báo cáo phân loại (trường hợp 2, RandomForest):")
print(classification_report(y_test_all, y_pred_all_rf))
```

Nhom5.ipynb

```
C: > Users > ngchn > Downloads > ptdfinal > Nhom5.ipynb > ...
+ Code + Markdown | ▶ Run All ⚡ Restart ⌛ Clear All Outputs | ⌂ Variables ⌂ Outline ... Python 3.11.2
D> v # Đánh giá hiệu suất
accuracy_all_rf = accuracy_score(y_test_all, y_pred_all_rf)
print("Accuracy trên tập huấn luyện (trường hợp 1): 1.0")
print("Accuracy trên tập kiểm tra (trường hợp 2, RandomForest): 0.992849846782431")
print("Báo cáo phân loại (trường hợp 2, RandomForest):")
print(classification_report(y_test_all, y_pred_all_rf))

[150] ✓ 0.3s
Accuracy trên tập huấn luyện (trường hợp 1): 1.0
Accuracy trên tập kiểm tra (trường hợp 2, RandomForest): 0.992849846782431
Báo cáo phân loại (trường hợp 2, RandomForest):
precision recall f1-score support
          0         0.99      1.00      0.99      894
          1         0.99      0.93      0.96      85

   accuracy           0.99      0.96      0.98      979
   macro avg       0.99      0.96      0.98      979
weighted avg     0.99      0.99      0.99      979
```

+ Code + Markdown

```
plot_confusion_matrix(rf_all, y_test_all, y_pred_all_rf, 2)
```

[150] ✓ 0.1s

... Confusion Matrix - RandomForestClassifier (Trường hợp 2)

		0	1
0	893	6	
1	1	79	
True labels			
	Predicted labels		

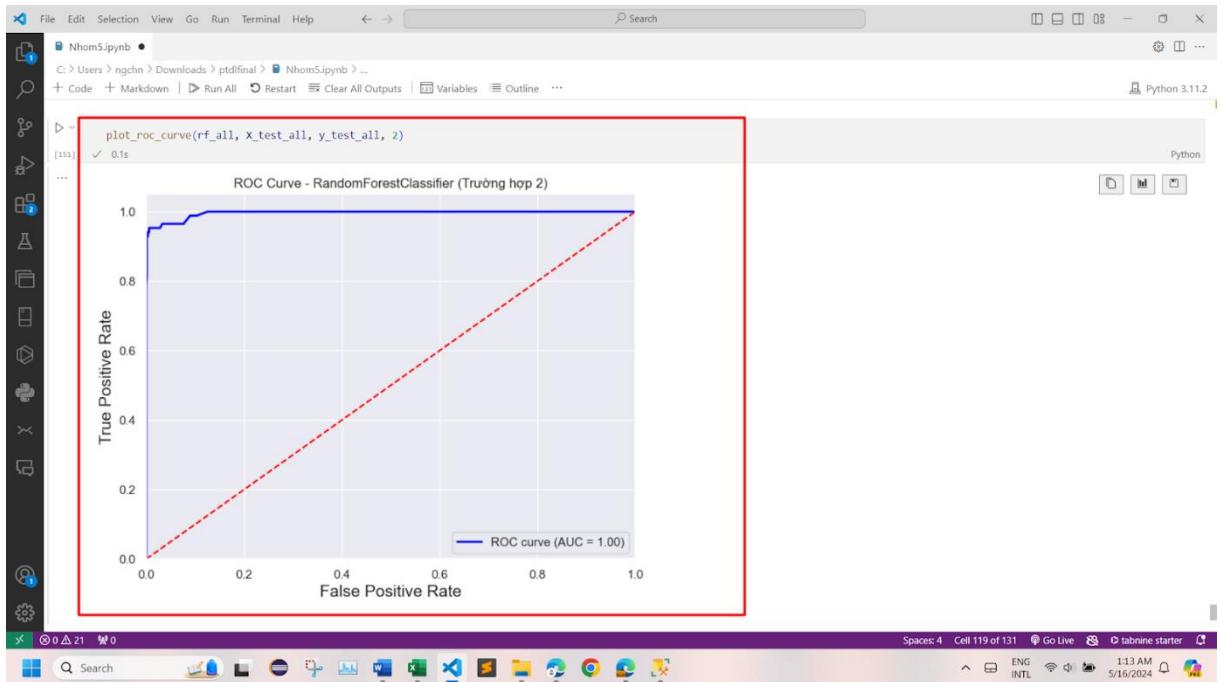
Spaces: 4 Cell 110 of 131 ⚡ Go Live ⌂ tabnine starter

Nhom5.ipynb

```
C: > Users > ngchn > Downloads > ptdfinal > Nhom5.ipynb > ...
+ Code + Markdown | ▶ Run All ⚡ Restart ⌛ Clear All Outputs | ⌂ Variables ⌂ Outline ... Python 3.11.2
D> v plot_confusion_matrix(rf_all, y_test_all, y_pred_all_rf, 2)
[150] ✓ 0.1s
... Confusion Matrix - RandomForestClassifier (Trường hợp 2)
```

		0	1
0	893	6	
1	1	79	
True labels			
	Predicted labels		

Spaces: 4 Cell 110 of 131 ⚡ Go Live ⌂ tabnine starter



Nhận xét: Random Forest Model khi dùng tất cả các thuộc tính thì cho ra kết quả dự báo chính xác hơn khi dùng các cột có tương quan cao

# CHƯƠNG 3: ĐÁNH GIÁ MÔ HÌNH VÀ ĐUẤA RA KẾT LUẬN SAU KHI PHÂN TÍCH

## 3.1. Đánh giá mô hình

- Tính toán các chỉ số đánh giá hiệu suất cho một mô hình máy học đã được huấn luyện dựa trên test data. Sau đó thực hiện hàm cho các mô hình học máy mà nhóm đã thực hiện.
- Đầu tiên nhóm sẽ đánh giá tổng quan trên TH2

The screenshot shows a Jupyter Notebook workspace titled 'Untitled (Workspace)'. The left sidebar displays a tree view of files and notebooks, including 'NLP-clustering-word--Vietnamese-S...', 'BankLoanStatus-DataAnalysis.ipynb', 'KiemTra.ipynb', 'Bank-Loan.ipynb', and 'FinalProject.ipynb'. The main area contains a cell with the following Python code:

```
def metrics_calculator(clf, X_test, y_test, model_name):
    y_pred = clf.predict(X_test)
    result = pd.DataFrame(data=[accuracy_score(y_test, y_pred),
                                precision_score(y_test, y_pred, average='binary'),
                                recall_score(y_test, y_pred, average='binary'),
                                f1_score(y_test, y_pred, average='binary'),
                                roc_auc_score(y_test, clf.predict_proba(X_test)[:,1])],
                           index=['Accuracy', 'Precision', 'Recall', 'F1-score', 'AUC'],
                           columns=[model_name])
    result = (result * 100).round(2).astype(str) + '%'
    return result
```

Below this cell, another cell is partially visible with the following code:

```
dt_result = metrics_calculator(dt_all, X_test_all, y_test_all, 'Decision Tree')
logreg_result = metrics_calculator(logreg_all, X_test_all, y_test_all, 'Logistic Regression')
rf_result = metrics_calculator(rf_all, X_test_all, y_test_all, 'Random Forest')
knn_result = metrics_calculator(knn_all, X_test_all, y_test_all, 'KNeighbors')
xgb_result = metrics_calculator(xgb_all, X_test_all, y_test_all, 'XGBoost')
```

- Vẽ mô hình F1-Score của các loại mô hình (sử dụng tất cả các biến)

File Edit Selection View Go Run Terminal Help

Untitled (Workspace)

Nhom5final.ipynb

Code Markdown Run All Restart Clear All Outputs Variables Outline

```
# Concatenate all results into a single DataFrame for comparison
all_results = pd.concat([dt_result, logreg_result, rf_result, kmm_result, xgb_result], axis=1)

# Extract the F1-scores
f1_scores = all_results.loc['F1-score']

# Convert the F1-scores to numeric values (they are currently strings with '%')
f1_scores_numeric = f1_scores.str.replace('%', '').astype(float)

# Plotting the F1-scores as a horizontal bar chart
plt.figure(figsize=(10, 6))
ax = f1_scores_numeric.plot(kind='barh', color='skyblue')
plt.title('F1-scores of Different Models use all fields')
plt.xlabel('F1-score (%)')
plt.ylabel('Model')
plt.xlim(0, 100) # Setting the x-axis limits for better visualization
plt.grid(axis='x', linestyle='--', linewidth=0.7)

# Annotate bars with F1-score values
for index, value in enumerate(f1_scores_numeric):
    plt.text(value + 1, index, f'{value:.2f}%', va='center')

# Show plot
plt.tight_layout()
plt.show()
```

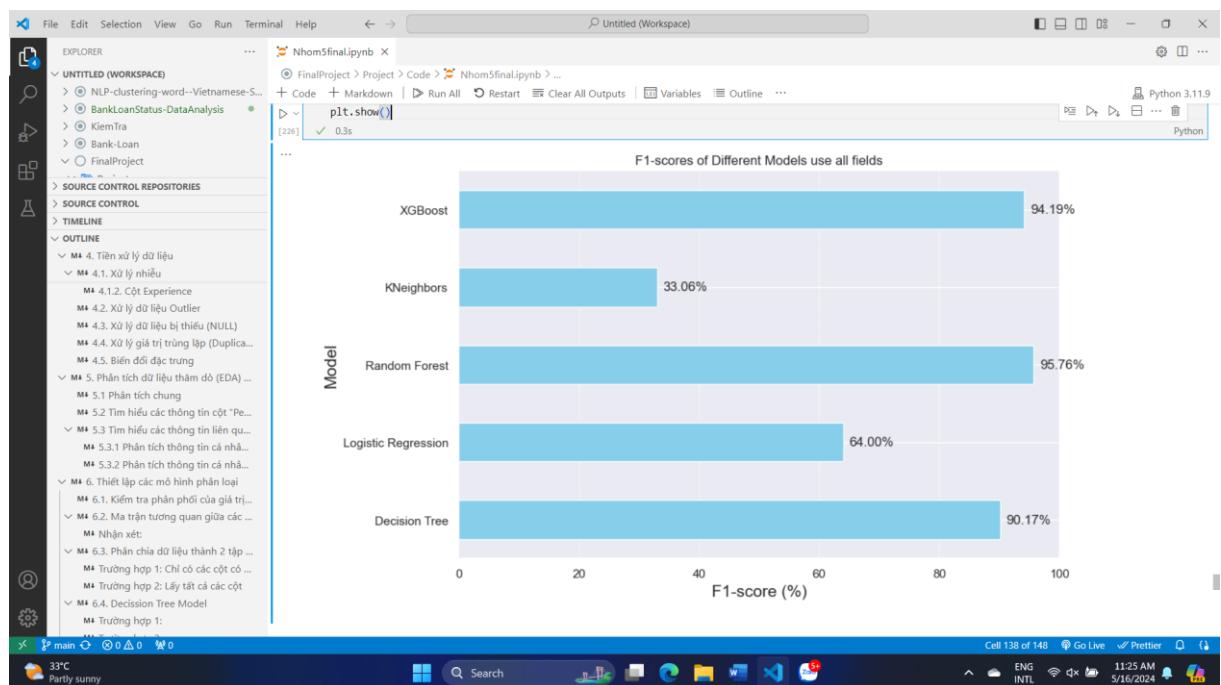
0.3s

F1-scores of Different Models use all fields

Model	F1-score (%)
XGBoost	94.19%
KNeighbors	33.06%
Random Forest	95.76%
Logistic Regression	64.00%
Decision Tree	90.17%

Spaces: 4 Cell 138 of 148 Go Live Prettier

11:24 AM 5/16/2024 ENG US



- Từ biểu đồ ta có thể thấy mô hình Random Forest là mô hình có chỉ số F1-Score tốt nhất trong 5 mô hình.
- Tiếp theo, nhóm tiếp tục đánh giá F1-Score cho các loại mô hình ở TH1

File Edit Selection View Go Run Terminal Help

Nhom5final.ipynb X

Untitled (Workspace)

Code + Markdown Run All Restart Clear All Outputs Variables Outline Python 3.11.9

• Tiếp tục vẽ cho TH1

```

dt_result = metrics_calculator(dt_selected, X_test_selected, y_test_selected, 'Decision Tree')
logreg_result = metrics_calculator(logreg_selected, X_test_selected, y_test_selected, 'Logistic Regression')
rf_result = metrics_calculator(rf_selected, X_test_selected, y_test_selected, 'Random Forest')
knn_result = metrics_calculator(knn_selected, X_test_selected, y_test_selected, 'KNeighbors')
xgb_result = metrics_calculator(xgb_selected, X_test_selected, y_test_selected, 'XGBoost')

selected_results = pd.concat([dt_result, logreg_result, rf_result, knn_result, xgb_result], axis=1)

# F1-scores
f1_scores = selected_results.loc['F1-score']

# Chuyển đổi F1-scores sang giá trị numeric values
f1_scores_numeric = f1_scores.str.replace('%', '').astype(float)

# Vẽ F1-scores
plt.figure(figsize=(10, 6))
ax = f1_scores_numeric.plot(kind='barh', color='skyblue')
plt.title('F1-scores của các loại mô hình (TH1)')
plt.xlabel('F1-score (%)')
plt.ylabel('Model')
plt.xlim(0, 100)
plt.grid(axis='x', linestyle='--', linewidth=0.7)

# Chú thích các thanh với giá trị F1-score
for index, value in enumerate(f1_scores_numeric):
    plt.text(value + 1, index, f'{value:.2f}%', va='center')

plt.tight_layout()
plt.show()

```

Spaces: 4 Cell 141 of 149 Go Live Prettier

33°C Party sunny

File Edit Selection View Go Run Terminal Help

Nhom5final.ipynb X

Untitled (Workspace)

Code + Markdown Run All Restart Clear All Outputs Variables Outline Python 3.11.9

• Tiếp tục vẽ cho TH1

```

dt_result = metrics_calculator(dt_selected, X_test_selected, y_test_selected, 'Decision Tree')
logreg_result = metrics_calculator(logreg_selected, X_test_selected, y_test_selected, 'Logistic Regression')
rf_result = metrics_calculator(rf_selected, X_test_selected, y_test_selected, 'Random Forest')
knn_result = metrics_calculator(knn_selected, X_test_selected, y_test_selected, 'KNeighbors')
xgb_result = metrics_calculator(xgb_selected, X_test_selected, y_test_selected, 'XGBoost')

selected_results = pd.concat([dt_result, logreg_result, rf_result, knn_result, xgb_result], axis=1)

# F1-scores
f1_scores = selected_results.loc['F1-score']

# Chuyển đổi F1-scores sang giá trị numeric values
f1_scores_numeric = f1_scores.str.replace('%', '').astype(float)

# Vẽ F1-scores
plt.figure(figsize=(10, 6))
ax = f1_scores_numeric.plot(kind='barh', color='skyblue')
plt.title('F1-scores của các loại mô hình (TH1)')
plt.xlabel('F1-score (%)')
plt.ylabel('Model')
plt.xlim(0, 100)
plt.grid(axis='x', linestyle='--', linewidth=0.7)

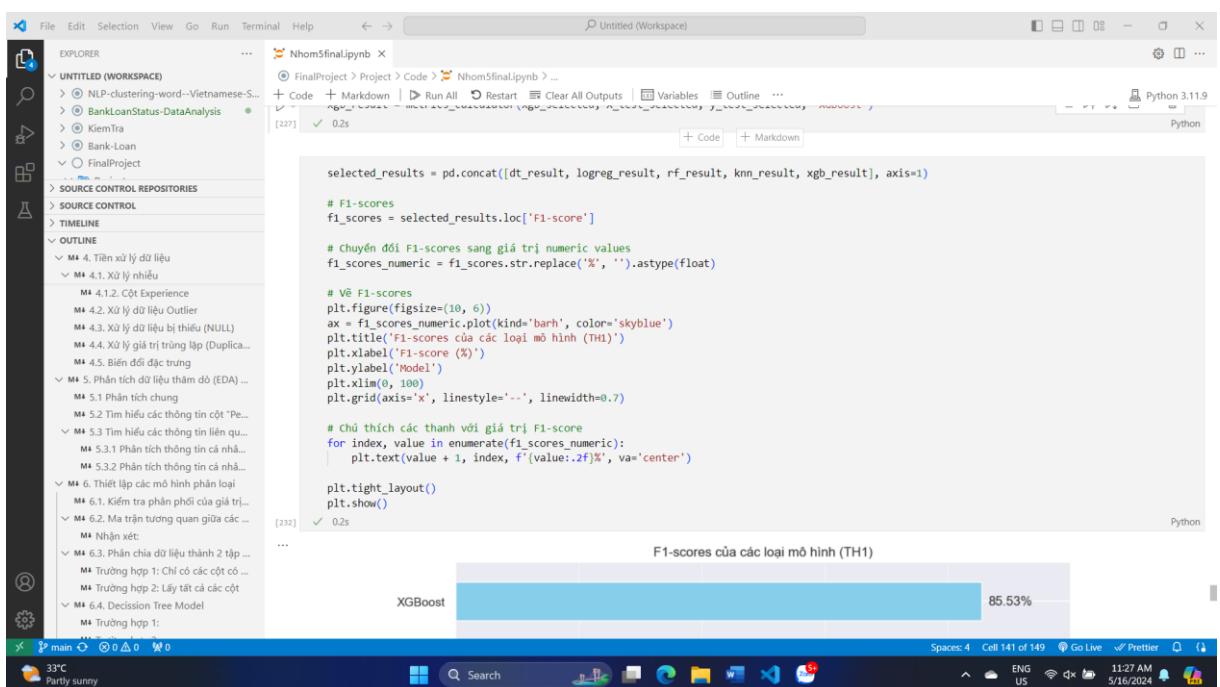
# Chú thích các thanh với giá trị F1-score
for index, value in enumerate(f1_scores_numeric):
    plt.text(value + 1, index, f'{value:.2f}%', va='center')

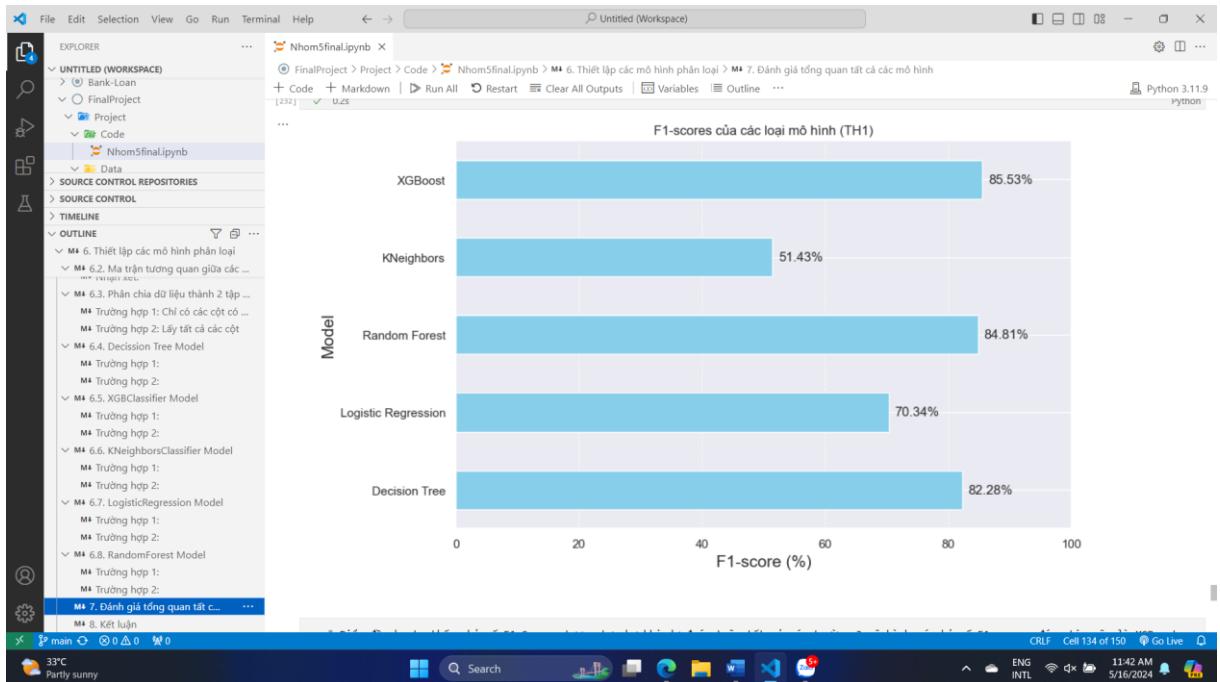
plt.tight_layout()
plt.show()

```

Spaces: 4 Cell 141 of 149 Go Live Prettier

33°C Party sunny





- Biểu đồ cho ta thấy chỉ số F1-Scores tương tự như khi dự đoán trên tất cả các trường 3 mô hình có chỉ số F1-scores đáng tin cậy là XGBoost, Random Forest và Decision Tree trong đó XGBoost vẫn là mô hình tốt nhất.

### 3.2. Kết luận sau khi phân tích

Model	Accuracy	Trường hợp 1							
		Precision		Recall		F1-score		Confusion matrix	
		0	1	0	1	0	1		
Decission Tree	0.971	0.98	0.89	0.99	0.76	0.98	0.82	[[886 8]	[20 65]]
XGB Classifier	0.976	0.98	0.92	0.99	0.8	0.99	0.86	[[888 6]	[17 68]]
KNeighbors Classifier	0.93	0.95	0.65	0.98	0.42	0.96	0.51	[[875 19]	[49 36]]
Logistic Regression	0.956	0.96	0.85	0.99	0.6	0.98	0.7	[[885 9]	[34 51]]
Random Forest	0.975	0.98	0.92	0.99	0.79	0.99	0.85	[[888 6]	[18 67]]

Model	Accuracy	Trường hợp 2							
		Precision		Recall		F1-score		Confusion matrix	
		0	1	0	1	0	1		
Decission Tree	0.982	0.99	0.89	0.99	0.92	0.99	0.9	[[884 10]	[7 78]]
XGB Classifier	0.989	1	0.93	0.99	0.95	0.99	0.94	[[888 6]	[4 81]]
KNeighbors Classifier	0.917	0.93	0.56	0.98	0.24	0.96	0.33	[[878 16]	[65 20]]
Logistic Regression	0.944	0.96	0.74	0.98	0.56	0.97	0.64	[[877 17]	[37 48]]
Random Forest	0.992	0.99	0.99	1	0.93	1	0.96	[[893 1]	[6 79]]

Trường hợp 1		Trường hợp 2	
Model	F1-Score	Model	F1-Score
Decission Tree	82.28%	Decission Tree	90.17%
XGB Classifier	85.53%	XGB Classifier	94.19%
KNeighbors Classifier	51.43%	KNeighbors Classifier	33.06%
Logistic Regression	70.34%	Logistic Regression	64.00%
Random Forest	84.81%	Random Forest	95.76%

- Với tập dữ liệu này thì các mô hình đạt kết quả dự đoán tốt khi sử dụng toàn bộ tập dữ liệu để dự đoán. Với chỉ số F1-Score đạt tới 95% (của Random forest)
- Decission Tree/XGB Classifier và Random Forest đều có độ chính xác cao khi chạy mô hình đạt 98%.

- Xét về chỉ số F1-score thi tỷ lệ % của Random Forest và XGB Classifier cao nên có thể lựa 1 trong 2 mô hình này để áp dụng dự đoán.
- Kết hợp với việc phân tích dữ liệu ban đầu (EDA) để nhận diện được những khách hàng có tiềm năng cho vay cá nhân.
- Trong tương lai nêu tập trung vào những người có xu hướng quan tâm đến vay cá nhân như:
  - Dựa vào độ tuổi: Tỷ lệ những người còn cao tuổi hoặc khoảng xung quanh 40 tuổi.
  - Dựa vào độ tuổi và kinh nghiệm làm việc: Những người có độ tuổi trung niên trở đi và có kinh nghiệm khoảng 10 năm trong công việc.
  - Dựa vào số lượng thành viên trong gia đình và trình độ học vấn: Những người có số lượng thành viên trong gia đình cao và trình độ học vấn cao.
  - Dựa vào yếu tố có tài khoản ngân hàng: Những người đã có tham gia gửi tiền tại ngân hàng.
  - Dựa vào chi tiêu: Những người có lượng chi tiêu trung bình càng cao.

Chính vì vậy ngân hàng nên xây dựng chính sách/chương trình/quyền lợi cho các đối tượng trên. Đây chính là các đối tượng tiềm năng để thu hút được nhiều khách hàng có tiềm năng tham gia khoản vay cá nhân.

### **3.3. Xây dựng một số chương trình cho các đối tượng có xu hướng quan tâm đến vay cá nhân.**

Nhóm tìm hiểu và xây dựng được một số chương trình, chính sách như sau:

1. Chương trình vay lãi suất ưu đãi cho khách hàng trung niên và cao tuổi:

- Mục tiêu: Khách hàng có độ tuổi từ 40 trở lên, đặc biệt là những người đã về hưu.
- Chính sách: Cung cấp các gói vay với lãi suất ưu đãi và điều kiện vay linh hoạt hơn. Có thể kết hợp với bảo hiểm sức khỏe hoặc các dịch vụ tài chính khác nhằm tạo sự hấp dẫn.

2. Chương trình vay hỗ trợ sự nghiệp cho khách hàng có kinh nghiệm làm việc:

- Mục tiêu: Khách hàng trung niên và có kinh nghiệm làm việc từ 10 năm trở lên.
- Chính sách: Các gói vay với điều kiện tốt, như lãi suất thấp, thời gian trả nợ dài hơn, hoặc miễn phí một số loại phí. Đặc biệt nhấn mạnh đến việc hỗ trợ các kế hoạch phát triển cá nhân hoặc khởi nghiệp.

3. Chương trình vay dành cho gia đình có nhiều thành viên và trình độ học vấn cao:

- Mục tiêu: Khách hàng có gia đình đông thành viên và có trình độ học vấn cao.
- Chính sách: Gói vay có ưu đãi dành cho các gia đình, như lãi suất thấp hơn, ưu đãi khi vay để mua sắm, sửa chữa nhà cửa, hoặc đầu tư vào giáo dục cho con cái.

4. Chương trình ưu đãi cho khách hàng đã có tài khoản ngân hàng:

- Mục tiêu: Khách hàng đã có tài khoản tiết kiệm hoặc giao dịch thường xuyên tại ngân hàng.
- Chính sách: Cung cấp các gói vay với lãi suất ưu đãi cho những khách hàng đã có lịch sử giao dịch tốt hoặc sở hữu tài khoản cao. Tặng điểm thưởng hoặc ưu đãi khi sử dụng các dịch vụ khác của ngân hàng.

5. Chương trình vay cho khách hàng có mức chi tiêu cao:

- Mục tiêu: Khách hàng có mức chi tiêu trung bình hàng tháng cao.
- Chính sách: Các gói vay với hạn mức cao, lãi suất cạnh tranh, và các ưu đãi khi sử dụng thẻ tín dụng hoặc các dịch vụ thanh toán khác của ngân hàng.

6. Chương trình tài chính cá nhân toàn diện:

- Mục tiêu: Khách hàng có nhu cầu vay cho nhiều mục đích khác nhau.
- Chính sách: Xây dựng các gói tài chính cá nhân linh hoạt, cho phép khách hàng vay cho nhiều mục đích khác nhau như tiêu dùng, mua sắm, du lịch, học tập... với các điều kiện và ưu đãi tùy chỉnh.

7. Chương trình hỗ trợ tài chính giáo dục:

- Mục tiêu: Khách hàng có con cái đang trong độ tuổi đi học.
- Chính sách: Các gói vay với lãi suất thấp hoặc miễn lãi suất trong năm đầu tiên cho các khoản vay phục vụ chi phí giáo dục, từ học phí đến các chi phí khác liên quan đến học tập.

8. Chương trình khách hàng thân thiết và tích điểm:

- Mục tiêu: Tất cả các đối tượng khách hàng.
- Chính sách: Tích điểm khi sử dụng dịch vụ của ngân hàng, từ đó đổi điểm lấy các ưu đãi về lãi suất, phí dịch vụ, hoặc các phần quà giá trị.

## **PHẦN 3: KẾT LUẬN**

### **1. KẾT QUẢ ĐẠT ĐƯỢC**

#### **1.1. Về kiến thức**

Nhóm năm được các kiến thức cũng như những vấn đề liên quan về các sử dụng các thư viện trên python để phân tích dữ liệu. Áp dụng kiến thức để thực hiện phân tích và áp dụng mô hình trên một tập dữ liệu cụ thể.

#### **1.2. Về việc thực hiện dự án**

Sử dụng thư viện trên python để có thể áp dụng kiến thức đã học để tiền xử lí, phân tích dữ liệu thăm dò cũng như trực quan hóa và áp dụng các mô hình phân loại bằng thực hành trực tiếp trên tập dữ liệu

Bank\_Personal\_Loan\_Modelling.

Thực hiện dự án một cách nhanh chóng và hiểu rõ được nhiệm vụ trong từng task của dự án.

### **2. ƯU ĐIỂM, HẠN CHẾ CỦA ĐỀ TÀI**

#### **2.1. Ưu điểm**

Thực hiện được các thao tác tiền xử lý dữ liệu, phân tích thăm dò, trực quan hóa và áp dụng mô hình trên tập dữ liệu Bank\_Personal\_Loan\_Modelling để thực hiện dự đoán với độ chính xác cao.

#### **2.2. Hạn chế**

Chưa áp dụng được các mô hình học sâu nhằm tối ưu hơn kết quả dự đoán cho việc phân tích dữ liệu.

### **3. HƯỚNG PHÁT TRIỂN**

Trong tương lai, nhóm sẽ tiến hành áp dụng các mô hình học sâu vào dự án (như các mạng thần kinh nhân tạo FNN, CNN,...) nhằm chọn ra được mô hình tiên tiến nhất để có thể tạo mô hình đó áp dụng trên tập dữ liệu tương tự. Bên cạnh đó, nhóm tìm kiếm nhiều tập dữ liệu khác liên quan đến chủ đề Bank, nhằm tạo nên một cơ sở dữ liệu hoàn chỉnh về ngân hàng, từ đó tiến hành phân

tích cụ thể hơn trong từng tình huống, và xây dựng nên mô hình chatbox hỗ trợ người dùng trong lĩnh vực ngân hàng.

## TÀI LIỆU THAM KHẢO

- [1] Tài liệu các file PDF hướng dẫn về Data Analysis trên nhiều tập dữ liệu khác nhau của GVHD Ths. Nguyễn Văn Thành
- [2] OVNE-Giáo Dục Việt (no date). Máy Học Ứng Dụng - Bài 8. Phép Đánh Giá ROC-AUC - Khái Niệm Ngưỡng Xác Suất. Available at:  
[https://www.youtube.com/watch?v=MPIwEPFGaOw&list=PLVoa6AIS1id7P0dmX26a\\_b7cKMJTRvSGF&index=7](https://www.youtube.com/watch?v=MPIwEPFGaOw&list=PLVoa6AIS1id7P0dmX26a_b7cKMJTRvSGF&index=7). [Accessed 9 May 2024]
- [3] Machine Learning cho dữ liệu dạng bảng (no date). Phân tích Khám phá Dữ liệu – EDA. Available at:  
[https://machinelearningcoban.com/tablml\\_book/ch\\_data\\_processing/eda.html](https://machinelearningcoban.com/tablml_book/ch_data_processing/eda.html).  
[Accessed 1 May 2024]
- [4] Deep AI KhanhBlog (no date). AUC. Available at:  
[https://phamdinhkhanh.github.io/deepai-book/ch\\_ml/modelMetric.html#auc](https://phamdinhkhanh.github.io/deepai-book/ch_ml/modelMetric.html#auc).  
[Accessed 9 May 2024]