ĐẠI HỌC HUẾ
KHOA KỸ THUẬT VÀ CÔNG NGHỆ

BÁO CÁO ĐỒ ÁN
HỌC KÌ I, năm học 2023-2024

Học phần:
HỌC MÁY 1

Số phách (Do hội đồng chấm ghi thi)

Thừa Thiên Huế, tháng 12 năm 2023.

ĐẠI HỌC HUẾ
KHOA KỸ THUẬT VÀ CÔNG NGHỆ

BÁO CÁO ĐỒ ÁN
HỌC KÌ I, năm học 2022-2023

Học phần:
NGÔN NGỮ LẬP TRÌNH PYTHON

Giảng viên hướng dẫn: T.s Hoàng Hữu Trung.
Lớp: Khoa học dữ liệu và Trí tuệ nhân tạo khóa 3.
Sinh viên thực hiện: Phạm Phước Bảo Tín.
(ký và ghi rõ họ tên)

Số phách **(**Do hội đồng chấm ghi thi)

Thừa Thiên Huế, tháng 12 năm 2023.

LỜI CẢM ƠN

Được trở thành sinh viên Khoa Kỹ Thuật và Công Nghệ - Đại học Huế em rất hạnh phúc và biết ơn. Hạnh phúc vì mình đã đạt được mục tiêu mong muốn và biết ơn sự cống hiến, chỉ bảo tận tình sâu sắc của quý thầy cô trong khoa đồng thời đã tạo điều kiện học tập lí tưởng cho chúng em. Để hoàn thành đồ án một cách chỉnh chu nhất có thể em xin gửi lời cảm ơn đến thầy giáo bộ môn - TS. Hoàng Hữu Trung đã hướng dẫn tận tình, chi tiết cho chúng em trong quá trình hoàn thành đồ án lẫn quá trình học tập học kì đầu tiên của đời sinh viên chúng em. Hi vọng rằng thời gian sắp tới em sẽ luôn cố gắng, nổ lực hơn nữa trong học tập chuyên nghành của mình.

Trong quá trình hoàn thành đồ án mặc dù em đã chuẩn bị kĩ nhưng không thể tránh khỏi những sai sót, em mong nhận được sự góp ý từ quý thầy, cô. Lời cuối cùng em xin kính chúc quý thầy, cô thật nhiều sức khỏe để tiếp tục dẫn dắt chúng em và những thế hệ tiếp theo thành người.

DANH MỤC HÌNH ẢNH

MỤC LỤC

DANH MỤC BẢNG BIỂU

PHẦN 1: HỌC CÓ GIÁM SÁT (SUPERVISED LEARNING)

1.1 Giới thiệu dữ liệu và mục đích bài toán

1.1.1 Giới thiệu dữ liệu

Dữ liệu được dùng trong phần học có giám này là datasets về hiện tượng khách hàng chấm dứt hoặc tiếp tục đăng kí với nhà cung cấp dịch vụ. Hiểu được lí do dừng sử dụng dịch vụ của khách khách là yếu tố quan trọng giúp các nhà cung cấp dịch vụ phát triển các chiến lược để giữ chân khách hàng hiện tại và thu hút khách hàng tiềm năng.

Datasets về các yếu tố ảnh hưởng đến việc khách hàng tiếp tục hay ngừng sử dụng dịch vụ của nhà cung cấp:

- CustomerID: Mã khách hàng.
- Age: Tuổi khách hàng.
- Gender: Giới tính khách hàng.
- Tenure: Số lần kí hợp đồng.
- Usage Frequency: Số lần sử dụng.
- Support Calls: Số cuộc gọi hỗ trợ.
- Payment Delay: Số ngày trễ hạn thanh toán.
- Subscription Type: Loại đăng kí.
- Contract Lenght: Loại thanh toán (theo thàng, năm,..)
- Total Spend: Tổng chi tiêu trong quá trình sử dụng dịch vụ (usd)
- Last Interaction: Số ngày tương tác cuối cùng.
- Churn: Kết quả đã ngưng sử dụng dịch vụ hay chưa (0: tiếp tục,1: đã ngưng)

CustomerID	Age	Gender	Tenure	Usage Frequenc	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interactio	Churn
1	22	Female	25	14	4	27	Basic	Monthly	598	9	1
2	41	Female	28	28	7	13	Standard	Monthly	584	20	0
3	47	Male	27	10	2	29	Premium	Annual	757	21	0
4	35	Male	9	12	5	17	Premium	Quarterly	232	18	0
5	53	Female	58	24	9	2	Standard	Annual	533	18	0
6	30	Male	41	14	10	10	Premium	Monthly	500	29	0
7	47	Female	37	15	9	28	Basic	Quarterly	574	14	1
8	54	Female	36	11	0	18	Standard	Monthly	323	16	0
9	36	Male	20	5	10	8	Basic	Monthly	687	8	0
10	65	Male	8	4	2	23	Basic	Annual	995	10	0
11	46	Female	42	27	9	21	Standard	Annual	526	3	1
12	56	Male	13	23	5	14	Basic	Quarterly	187	1	0
13	31	Male	2	7	0	25	Premium	Quarterly	758	24	0
14	42	Male	46	27	5	8	Premium	Quarterly	438	30	0
15	59	Male	21	17	2	14	Premium	Quarterly	663	15	0
16	35	Female	1	3	7	3	Basic	Monthly	677	25	1
17	29	Male	54	3	6	2	Basic	Monthly	636	22	0
18	45	Male	9	30	4	25	Basic	Annual	127	18	0
19	65	Female	40	2	1	6	Premium	Annual	396	21	0
20	62	Male	39	19	2	15	Premium	Quarterly	202	24	0
21	48	Male	28	7	1	21	Premium	Monthly	925	13	0
22	36	Female	58	4	0	1	Premium	Quarterly	463	26	0
23	55	Male	50	28	0	17	Standard	Quarterly	449	3	0
24	36	Female	54	20	4	1	Basic	Monthly	373	25	0
25	64	Male	59	7	5	9	Basic	Annual	460	12	0
26	65	Female	58	7	3	30	Premium	Annual	166	1	1
27	53	Female	58	18	8	4	Basic	Quarterly	615	4	0
28	41	Female	60	7	7	0	Premium	Annual	696	20	0

Hình 1: Dữ liệu trước tiền xử lí

1.1.2 Mục đích bài toán

1.2 Tiền xử lí dữ liệu

Dữ liệu ban đầu gồm loại dữ liệu số và dữ liệu phân loại không nhất quán, để phục vụ cho việc trực quan hóa dữ liệu và thực hiện bài toán phân loại việc tiền xử lí dữ liệu đóng vai trò rất quan trọng.

1.2.1 Làm sạch dữ liệu

Công việc làm sạch dữ liệu là một phần của tiền xử lí dữ liệu, gồm những việc dưới đây:

1. Kiểm tra giá trị trùng lặp

   Sử dụng phương thức duplicated trong pandas để kiểm tra số hàng trùng lặp trong datasets, kết quả thu được như Hình 2.

```
# kiểm tra giá trị trùng lặp
df.duplicated().sum()
```

```
0
```

Hình 2: Kết quả kiểm tra giá trị trùng lặp

2. Xử lí giá trị thiếu

   Sử dụng phương thức isna để kiểm tra số giá trị thiếu trong mỗi cột có trong datasets, kết quả thu được như Hình 3.

```
# Tìm missing values
df.isna().sum()
```

```
CustomerID           0
Age                  0
Gender               0
Tenure               0
Usage Frequency      0
Support Calls        0
Payment Delay        0
Subscription Type    0
Contract Length      0
Total Spend          0
Last Interaction     0
Churn                0
dtype: int64
```

Hình 3: Kiểm tra giá trị thiếu

3. Loại bỏ cột không cần thiết

   Cột mã số khách hàng không có vai trò trong mô hình chúng tôi sẽ sử dụng vì thế có thể loại bỏ cột này để giảm tải dữ liệu, thu kết quả như Hình 4.

```
# loại bỏ các cột không cần thiết
df = df.drop(columns = 'CustomerID')
df
```

	Age	Gender	Tenure	Usage Frequency	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interaction	Churn
0	22	Female	25	14	4	27	Basic	Monthly	598	9	1
1	41	Female	28	28	7	13	Standard	Monthly	584	20	0
2	47	Male	27	10	2	29	Premium	Annual	757	21	0
3	35	Male	9	12	5	17	Premium	Quarterly	232	18	0
4	53	Female	58	24	9	2	Standard	Annual	533	18	0
...	...	...	...	...	...	...	...	...	...	...	...
64369	45	Female	33	12	6	21	Basic	Quarterly	947	14	1
64370	37	Male	6	1	5	22	Standard	Annual	923	9	1
64371	25	Male	39	14	8	30	Premium	Monthly	327	20	1
64372	50	Female	18	19	7	22	Standard	Monthly	540	13	1
64373	52	Female	45	15	9	25	Standard	Monthly	696	22	1

64374 rows × 11 columns

Hình 4: Loại bỏ cột không cần thiết

4. Xử lí giá trị nhiễu

Thống kê cơ bản các cột giá trị trong datasets như Hình 5.

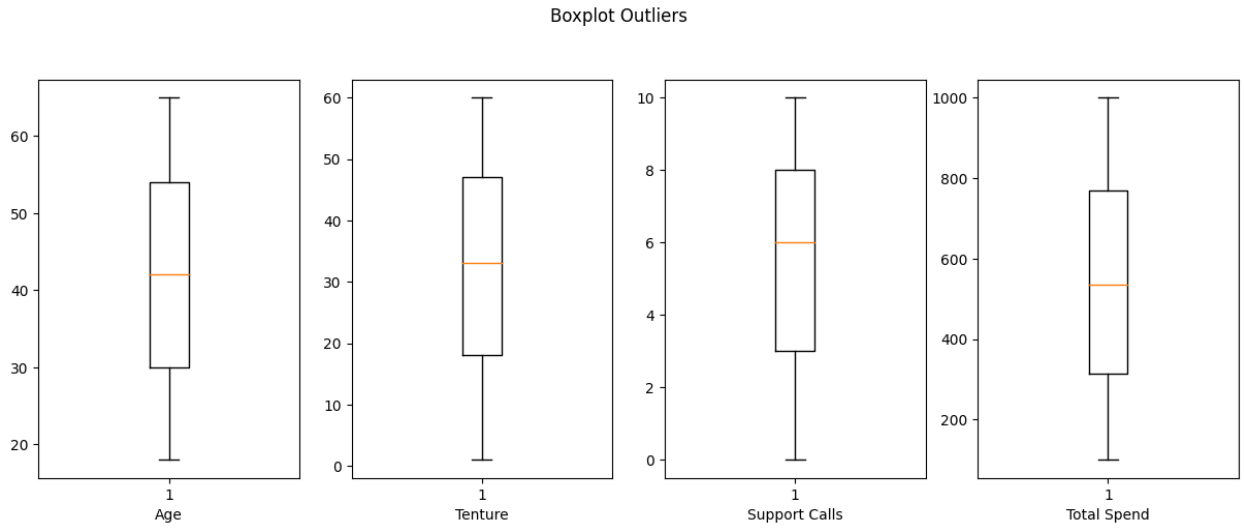
	CustomerID	Age	Tenure	Usage Frequency	Support Calls	Payment Delay	Total Spend	Last Interaction	Churn
count	64374.000000	64374.000000	64374.000000	64374.000000	64374.000000	64374.000000	64374.000000	64374.000000	64374.000000
mean	32187.500000	41.970982	31.994827	15.080234	5.400690	17.133952	541.023379	15.498850	0.473685
std	18583.317451	13.924911	17.098234	8.816470	3.114005	8.852211	260.874809	8.638436	0.499311
min	1.000000	18.000000	1.000000	1.000000	0.000000	0.000000	100.000000	1.000000	0.000000
25%	16094.250000	30.000000	18.000000	7.000000	3.000000	10.000000	313.000000	8.000000	0.000000
50%	32187.500000	42.000000	33.000000	15.000000	6.000000	19.000000	534.000000	15.000000	0.000000
75%	48280.750000	54.000000	47.000000	23.000000	8.000000	25.000000	768.000000	23.000000	1.000000
max	64374.000000	65.000000	60.000000	30.000000	10.000000	30.000000	1000.000000	30.000000	1.000000

Hình 5: Thống kê cơ bản các giá trị trong cột

Kiểm tra giá trị ngoại lai bằng biểu đồ hộp

```
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(15, 5))

fig.suptitle('Boxplot Outliers', y=1.02)

axes[0].boxplot(df['Age'])
axes[0].set_xlabel('Age')

axes[1].boxplot(df['Tenure'])
axes[1].set_xlabel('Tenture')

axes[2].boxplot(df['Support Calls'])
axes[2].set_xlabel('Support Calls')

axes[3].boxplot(df['Total Spend'])
axes[3].set_xlabel('Total Spend')
```

Hình 6: Mã nguồn biểu đồ hộp kiểm tra giá trị nhiễu



Hình 7: Biểu đồ hộp kiểm tra giá trị nhiễu

1.2.2 Chuyển đổi dữ liệu

1. Đánh giá thuộc tính độc lập (số và phân loại):
2. Chuyển đổi các thuộc tính phân loại thành dạng số:

   Từ kết quả của việc đánh giá thuộc tính độc lập ta thu được 3 cột cần chuyển đổi thuộc tính từ phân loại sang số như dưới đây.

```
# Chuyển đổi các thuộc tính categorical thành dạng số

# gioi tinh
df['Gender']=LabelEncoder().fit_transform(df['Gender'])

# loai dang ki
df['Subscription Type']=LabelEncoder().fit_transform(df['Subscription Type'])

# loai thanh toan
df['Contract Length']=LabelEncoder().fit_transform(df['Contract Length'])
```

Hình 8: Chuyển đổi dữ liệu từ dạng phân loại sang dạng số

3. Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là một bước quan trọng trong quá trình tiền xử lý dữ liệu, nhằm chuyển các đặc trưng định lượng (number) về cùng một thang đo chung, giúp cho việc biểu diễn dữ liệu dễ dàng và các mô hình phân tích nhất là các mô hình học máy (machine learning) hoạt động hiệu quả hơn.

Có hai loại chuẩn hóa dữ liệu thường dùng đó là Standardization (Z-Score Scaling) và Normalization (Min-Max Scaling). Lần này chúng tôi sử dụng phướng pháp Min-Max Scaling để chuẩn hóa dữ liệu.

```
# cac cot can chuan hoa
columns_to_normalize = ['Age', 'Tenure', 'Usage Frequency', 'Support Calls', 'Payment Delay', 'Total Spend', 'Last Interaction','Churn']
scaler = MinMaxScaler()
df[columns_to_normalize] = scaler.fit_transform(df[columns_to_normalize])
```

Hình 9: Chuẩn hóa dữ liệu bằng loại Min-Max Scaling

Kết quả sau quá trình chuyển đổi và chuyển hóa dữ liệu đồng thời cũng là kết quả của quá trình tiền xử lí dữ liệu như Hình 10.

	Age	Gender	Tenure	Usage Frequency	Support Calls	Payment Delay	Subscription Type	Contract Length	Total Spend	Last Interaction	Churn
0	0.085106	0	0.406780	0.448276	0.4	0.900000	0	1	0.553333	0.275862	1.0
1	0.489362	0	0.457627	0.931034	0.7	0.433333	2	1	0.537778	0.655172	0.0
2	0.617021	1	0.440678	0.310345	0.2	0.966667	1	0	0.730000	0.689655	0.0
3	0.361702	1	0.135593	0.379310	0.5	0.566667	1	2	0.146667	0.586207	0.0
4	0.744681	0	0.966102	0.793103	0.9	0.066667	2	0	0.481111	0.586207	0.0
...	...	...	...	...	...	...	...	...	...	...	...
64369	0.574468	0	0.542373	0.379310	0.6	0.700000	0	2	0.941111	0.448276	1.0
64370	0.404255	1	0.084746	0.000000	0.5	0.733333	2	0	0.914444	0.275862	1.0
64371	0.148936	1	0.644068	0.448276	0.8	1.000000	1	1	0.252222	0.655172	1.0
64372	0.680851	0	0.288136	0.620690	0.7	0.733333	2	1	0.488889	0.413793	1.0
64373	0.723404	0	0.745763	0.482759	0.9	0.833333	2	1	0.662222	0.724138	1.0

64374 rows × 11 columns

Hình 10: Kết quả tiền xử lí dữ liệu

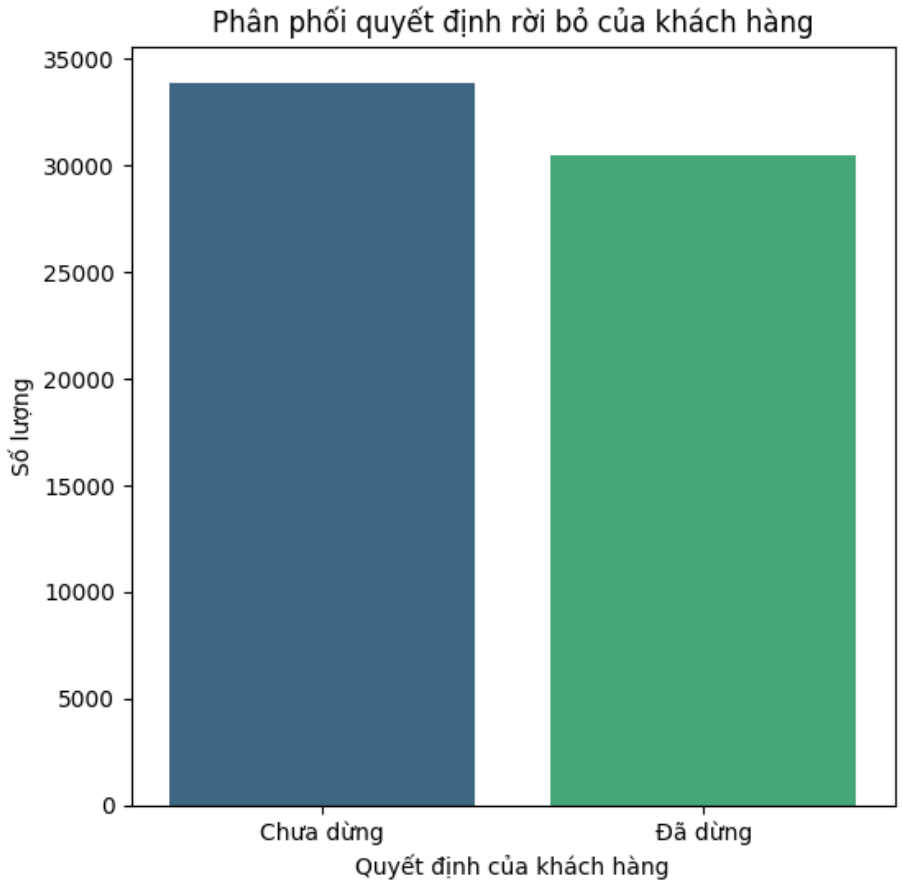
1.3 Trực quan hóa dữ liệu trước phân tích dữ liệu

1.2.1 Biểu đồ cột

Để cho nhà cung cấp nhìn ra được tổng quan trong việc khách hàng đã đưa ra quyết định có tiếp tục hay ngừng sử dụng dịch vụ của họ hay không như Hình 11.

```
plt.figure(figsize=(6, 6))
sns.countplot(data=df, x='Churn', palette='viridis')
plt.xlabel('Quyết định của khách hàng')
plt.ylabel('Số lượng')
plt.title('Phân phối quyết định rời bỏ của khách hàng')
plt.xticks([0, 1], ['Chưa dừng', 'Đã dừng'])
plt.show()
```

Hình 11: Mã nguồn biểu đồ cột



Hình 12: Kết quả biểu đồ cột

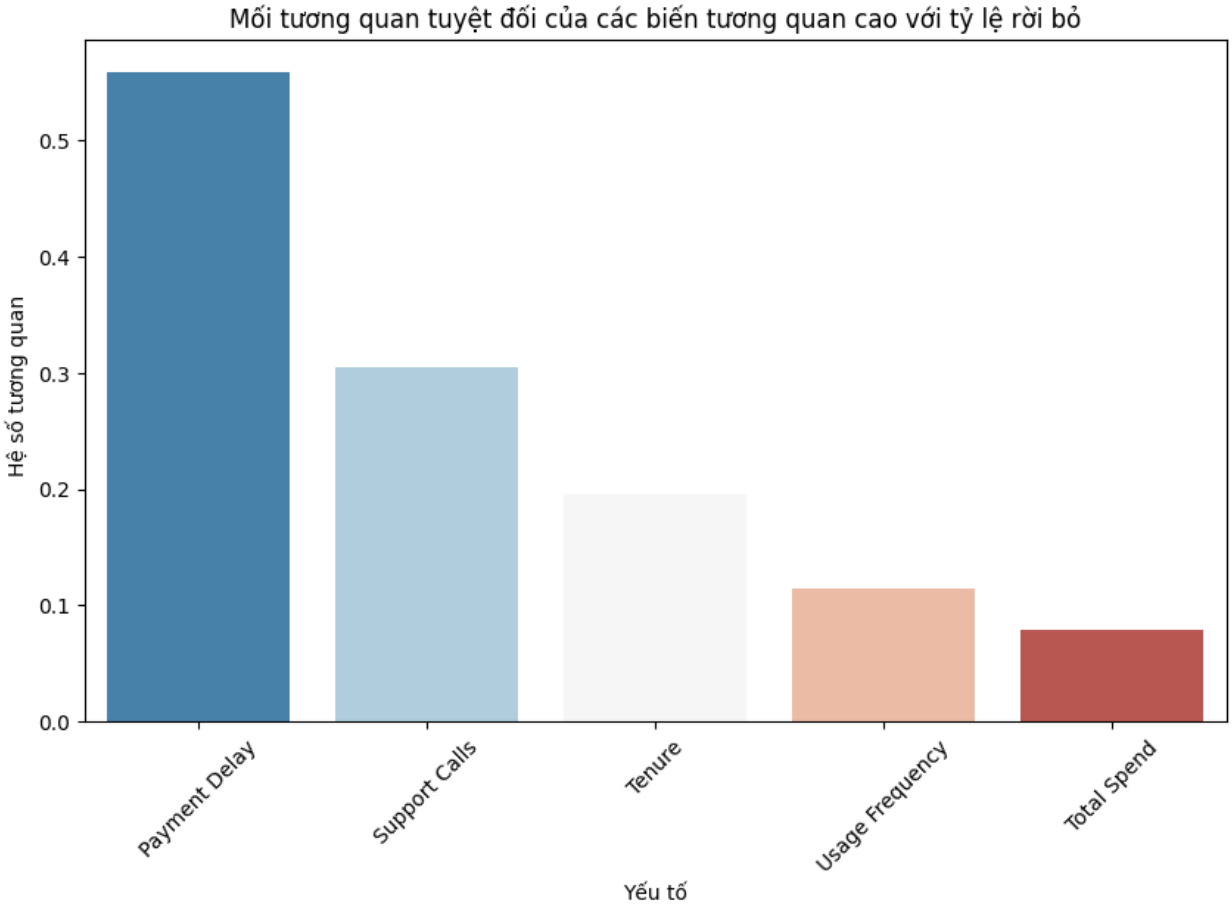
Kết quả như Hình 12 cho thấy việc số lượng khách hàng quyết định tiếp tục sử dụng nhiều hơn số khách hàng đã rời bỏ nhưng không chênh lệch đáng kể, đây cũng là một điều lo ngại khi số người hủy gần bằng người ở lại.

Trong phần này sử dụng biểu đồ cột để so sánh tương quan giữa các yếu tố với quyết định dừng sử dụng dịch vụ của khách hàng.

```
corr_matrix = df.corr()
high_corr_vars = corr_matrix.abs().nlargest(6, 'Churn')['Churn'].index[1:]
high_corr_values = corr_matrix.abs().nlargest(6, 'Churn')['Churn'].values[1:]
plt.figure(figsize=(10, 6))
sns.barplot(x=high_corr_vars, y=high_corr_values, palette='RdBu_r')
plt.ylabel('Hệ số tương quan ')
plt.xlabel('Yếu tố')
plt.title('Mối tương quan tuyệt đối của các biến tương quan cao với tỷ lệ rời bỏ')
plt.xticks(rotation=45)
plt.show()
```

Hình 13: Biểu đồ cột về hệ số tương quan

Kết quả:



Hình 14: Kết quả biểu đồ cột hệ số tương quan

Nhận được kết quả như , chúng ta thấy được rằng yếu tố "Payment delay" (số ngày trễ hạn thanh toán) có môi tương quan khá cao với tỉ lệ khách hàng rời bỏ nhà cung cấp.

1.2.2 Biểu đồ đường

Nhà cung cấp cần nắm được nguồn thu của mình của các nhóm tuổi để có chiến lược phát triển kinh doanh.

```
total_spend_by_age = df.groupby('Age')['Total Spend'].sum().reset_index()
# Vẽ biểu đồ đường
plt.figure(figsize=(10, 6))
plt.plot(total_spend_by_age['Age'], total_spend_by_age['Total Spend'], marker='.', linestyle='-', color='b')
plt.xlabel('Age')
plt.ylabel('Total Spend')
plt.title('Total Spend by Age')
plt.grid(True)
plt.show()
```

Hình 15: Mã nguồn biểu đồ đường

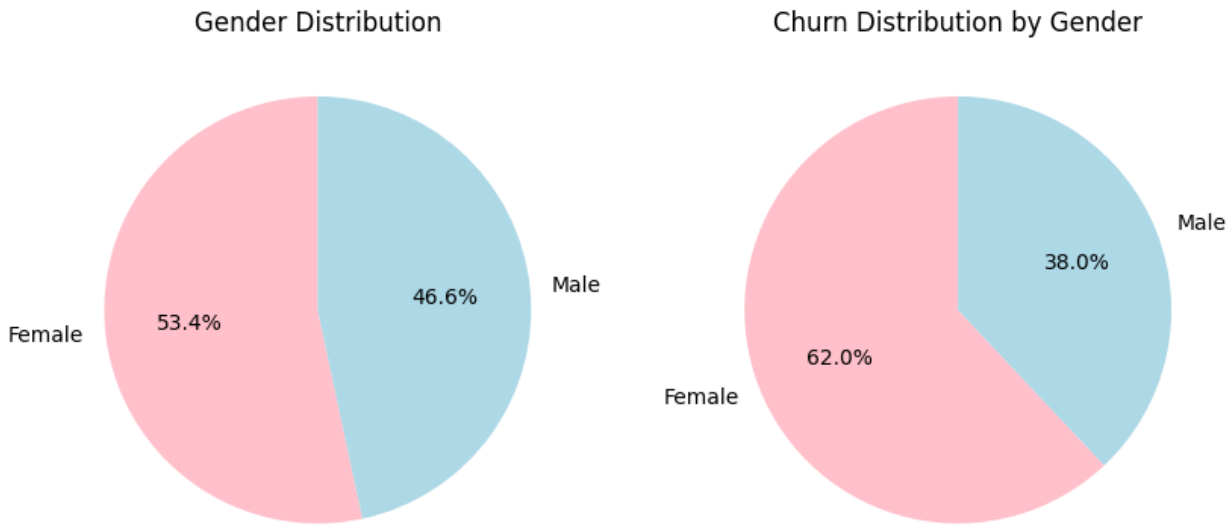
1.2.3 Biểu đồ tròn

Thống kê tỉ lệ giới tính phù hợp với việc sử dụng biểu đồ tròn, xem đối tượng nào là khách hàng chiếm trọng số, và đối tượng nào chiếm trọng số trong việc ngừng sử dụng dịch vụ doanh nghiệp.

```
# thống kê số lượng khách hàng nam và nữ
gender_counts=df['Gender'].value_counts()
# thống kê giới tính khách hàng đã ngưng sử dụng dịch vụ
gender_counts_1 = churn_1['Gender'].value_counts()
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 6))
# Biểu đồ tròn về phân phối giới tính
axes[0].pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90, colors=['pink', 'lightblue'])
axes[0].set_title('Gender Distribution')
# Biểu đồ tròn về tình trạng chấm dứt hợp đồng
axes[1].pie(gender_counts_1, labels=gender_counts_1.index, autopct='%1.1f%%', startangle=90, colors=['pink', 'lightblue'])
axes[1].set_title('Churn Distribution by Gender')
# Hiển thị biểu đồ
plt.show()
```

Hình 16: Mã nguồn biểu đồ tròn

Kết quả:



Hình 17: Kết quả biểu đồ tròn phân phối giới tính

Từ Hình 17 nhận thấy rằng tỉ lệ nam và nữ trong tệp những khách hàng đã đăng kí chênh lệch không nhiều. Ngược lại trong tệp khách hàng quyết định ngừng sử dụng dịch vụ tỉ lệ nữ và nam lệch nhau lớn, đây cũng là một yếu tố quan trọng ảnh hưởng đến mô hình.

1.2.4 Biểu đồ phân tán

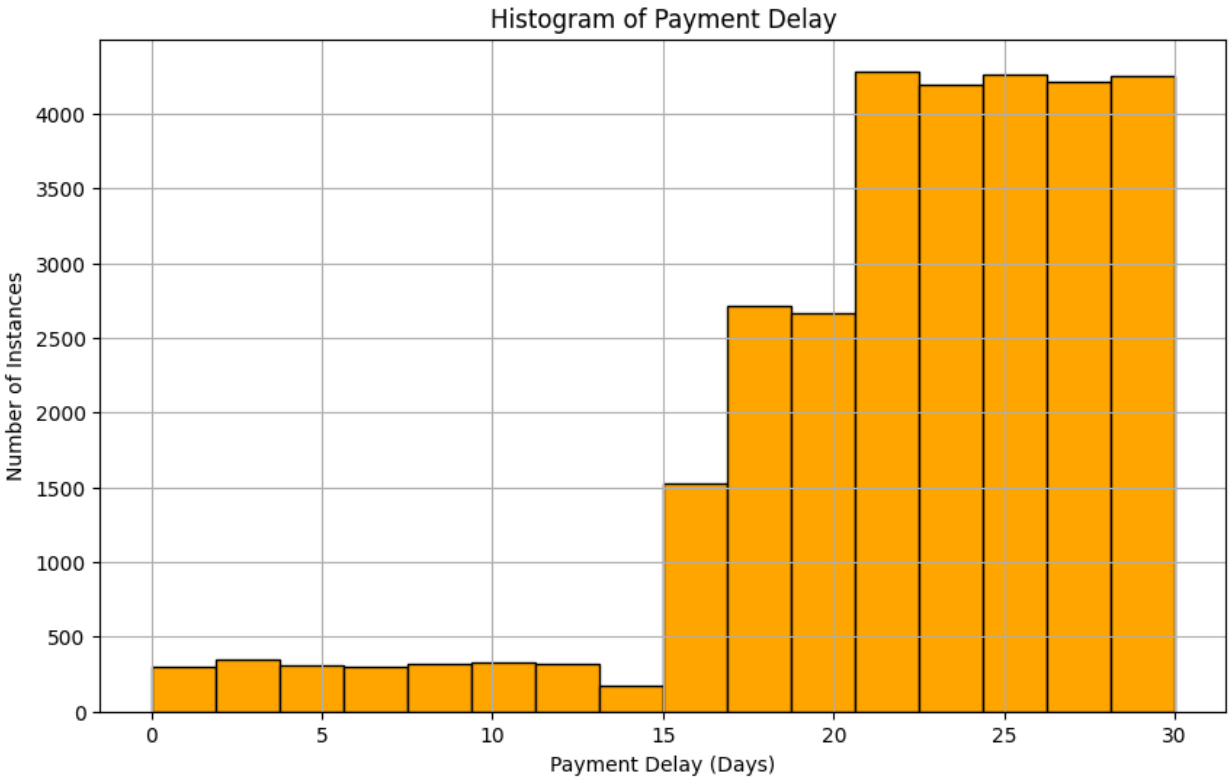
1.2.5 Biểu đồ phân phối tần suất

Nhận thấy kết quả hệ số tương quan giữa các yếu tố với quyết định rời bỏ của khách hàng từ Hình 14 thì yếu tố "Payment delay" có tương quan cao nhất. Biểu đồ phân phối tần suất dưới đây biểu thị số ngày trễ hạn dẫn đến việc không tiếp tục sử dụnh.

```
plt.figure(figsize=(10, 6))
plt.hist(churn_1['Payment Delay'], bins=16, color='orange', edgecolor='black')
plt.xlabel('Payment Delay (Days)')
plt.ylabel('Number of Instances')
plt.title('Histogram of Payment Delay')
plt.grid(True)
plt.show()
```

Hình 18: Mã nguồn biểu đồ tần suất

Kết quả:



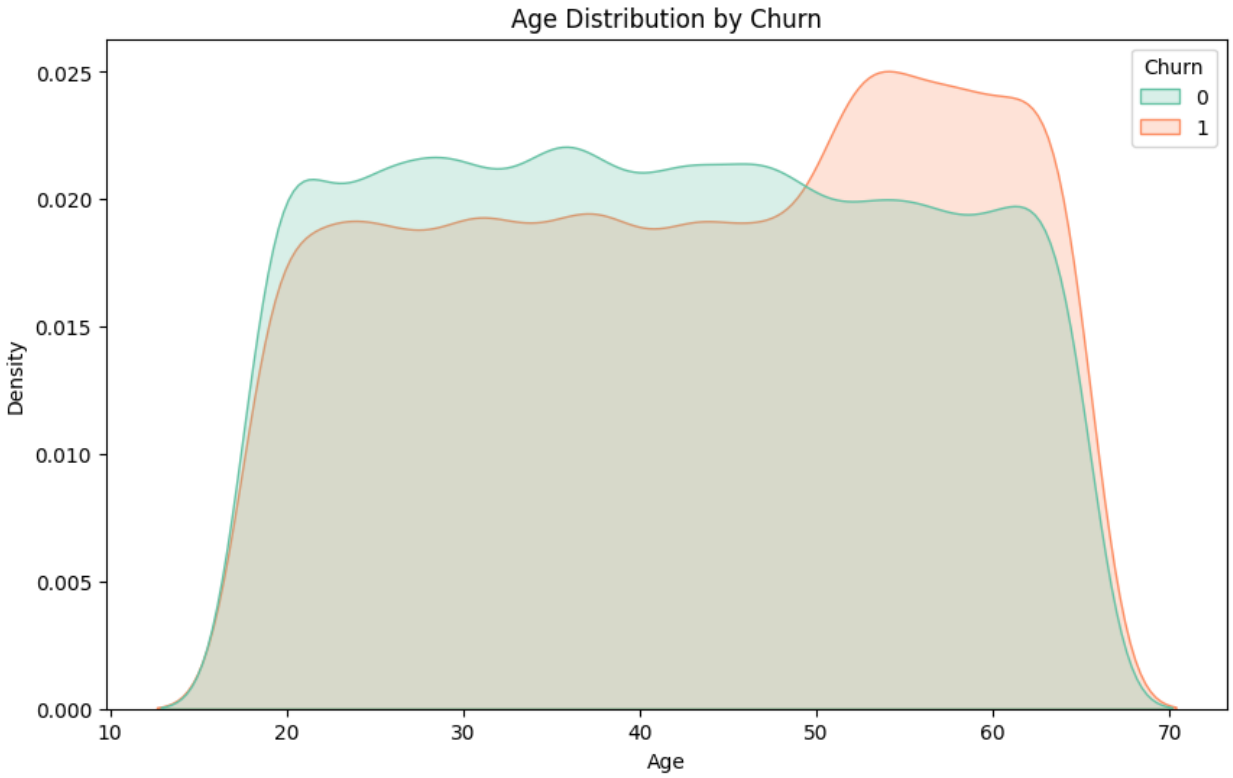
Hình 19: Biểu đồ tần suất

Nhóm người quyết định hủy dịch vụ đại đa số nằm trong khoảng thanh toán trễ lâu từ 20-30 ngày.

1.2.6 Biểu đồ mật độ

```
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df, x='Age', hue='Churn', common_norm=False, fill=True, palette='Set2')
plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Age Distribution by Churn')
plt.show()
```

Hình 20: Mã nguồn biểu đồ mật độ

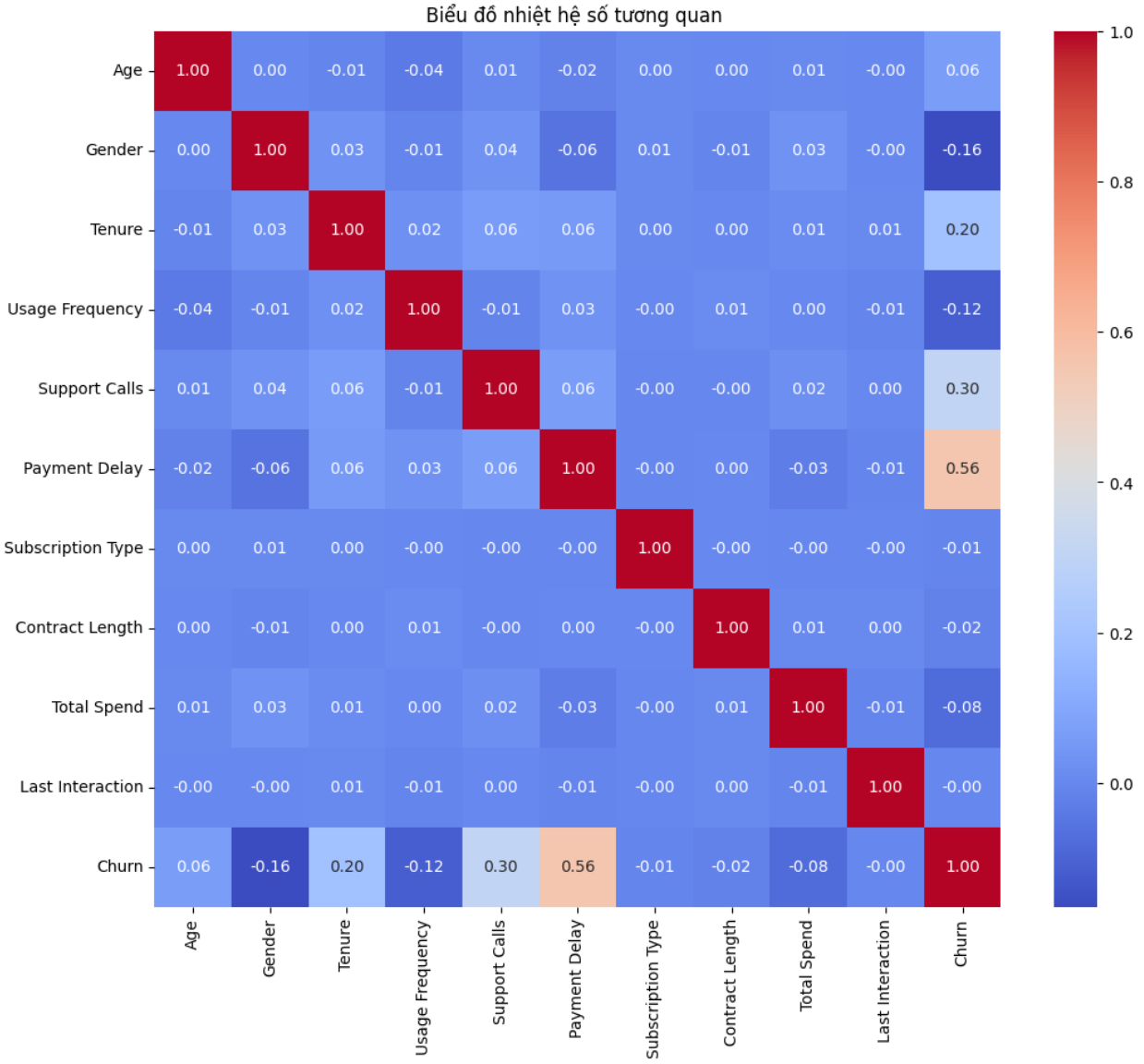


Hình 21: Biểu đồ mật độ

Các khoảng tuổi thuộc từ 20-50 đều có chênh lệch tỉ trọng ổn định, đồng đều qua các khoảng. Đặc biệt khoảng tuổi từ 55-65 độ chênh lệch tỉ trọng giữa việc tiếp tục và ngưng sử dụng lớn hơn nhiều so với các khoảng còn lại.

1.2.7 Biểu đồ nhiệt

Biểu đồ nhiệt (Heatmap) trích xuất các mối tương quan của các yếu tố đặc trưng với nhau, hoặc cột mục tiêu, từ đó có thể dựa vào biểu đồ để đưa ra các đặc vào tập huấn luyện và tập kiểm tra.



Hình 22: Biểu đồ nhiệt hệ số tương quan

Kết quả từ Hình 22, ta nhận thấy được các đặc trưng như "Payment Delay", "Support calls", "Tenure", "Usage Frequency" có tương quan với "Churn".

1.4 Ứng dụng các mô hình học giám sát vào bài toán

1.3.1 Mục đích bài toán

Trong bài toán, chúng em sử dụng mô hình thuật toán trong học có giám sát để ứng dụng dự đoán phân loại khách hàng quyết định ngừng sử dụng dịch vụ của nhà cung cấp hay không.

1.3.2 Thư viện Lazy Predict

Lazy Predict là một thư viện Python được sử dụng để tự động đánh giá hiệu suất của nhiều mô hình máy học khác nhau cho một tập dữ liệu cụ thể. Thư viện này tự động chọn và huấn luyện một loạt các mô hình khác nhau mà không cần bạn phải thủ công chỉ định từng mô hình cụ thể. Nó giúp tiết kiệm thời gian trong quá trình đánh giá và so sánh hiệu suất của các mô hình.

100%|██████████| 29/29 [05:09<00:00, 10.67s/it]

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score
LGBMClassifier	0.87	0.87	0.87	0.87
RandomForestClassifier	0.86	0.86	0.86	0.86
XGBClassifier	0.86	0.86	0.86	0.86
ExtraTreesClassifier	0.86	0.86	0.86	0.86
AdaBoostClassifier	0.86	0.86	0.86	0.86
SVC	0.85	0.86	0.86	0.85
BaggingClassifier	0.86	0.85	0.85	0.86
KNeighborsClassifier	0.85	0.85	0.85	0.85
DecisionTreeClassifier	0.85	0.85	0.85	0.85
ExtraTreeClassifier	0.85	0.85	0.85	0.85
NuSVC	0.83	0.83	0.83	0.83
GaussianNB	0.83	0.83	0.83	0.83
QuadraticDiscriminantAnalysis	0.82	0.83	0.83	0.82
LinearDiscriminantAnalysis	0.82	0.82	0.82	0.82
RidgeClassifier	0.82	0.82	0.82	0.82
RidgeClassifierCV	0.82	0.82	0.82	0.82
LinearSVC	0.82	0.82	0.82	0.82
SGDClassifier	0.82	0.82	0.82	0.82
CalibratedClassifierCV	0.82	0.82	0.82	0.82
LogisticRegression	0.81	0.81	0.81	0.81
NearestCentroid	0.81	0.81	0.81	0.81
Perceptron	0.79	0.79	0.79	0.79
BernoulliNB	0.78	0.78	0.78	0.78
PassiveAggressiveClassifier	0.79	0.78	0.78	0.78
DummyClassifier	0.53	0.50	0.50	0.36

Hình 23: Đánh giá hiệu suất mô hình bằng Lazy Preditct

Model	Time Taken
LGBMClassifier	0.30
RandomForestClassifier	3.39
XGBClassifier	0.27
ExtraTreesClassifier	2.76
AdaBoostClassifier	0.71
SVC	50.77
BaggingClassifier	0.45
KNeighborsClassifier	0.31
DecisionTreeClassifier	0.08
ExtraTreeClassifier	0.05
NuSVC	90.40
GaussianNB	0.05
QuadraticDiscriminantAnalysis	0.05
LinearDiscriminantAnalysis	0.20
RidgeClassifier	0.05
RidgeClassifierCV	0.05
LinearSVC	0.52
SGDClassifier	0.09
CalibratedClassifierCV	0.15
LogisticRegression	0.08
NearestCentroid	0.04
Perceptron	0.03
BernoulliNB	0.03
PassiveAggressiveClassifier	0.05
DummyClassifier	0.03

Hình 24: Thời gian thực hiện từng mô hình

Từ kết quả nhận được 3 mô hình có hiệu suất tốt nhất để thực hiện thủ công từng mô hình: [LGBMClassifier, RandomForestClassifier, XGBClassifier]

Chọn các đặc trưng có tương quan với mục tiêu để đưa vào mô hình.

```
selected_features = ["Payment Delay", "Support Calls", "Tenure", "Usage Frequency"]
data_subset = df[selected_features + ['Churn']]
features = data_subset.drop('Churn', axis=1)
target= data_subset['Churn']
```

Hình 25: Chọn các đặc trưng đưa vào mô hình

Dữ liệu được chia ngẫu nhiên thành tập huấn luyện chiếm 80% kích thước mẫu và tập kiểm thử chiếm 20% kích thước mẫu.

```
# Chia thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

Hình 26: Chia ngẫu nhiễn tập huấn luyện và tập kiểm tra

1.3.2 Mô hình Light Gradient Boosting Machine (LGBM)

1. Giới thiệu mô hình LGBM

LGBM (Light Gradient Boosting Machine) là một framework gradient boosting sử dụng các thuật toán học cây quyết định. Nó được thiết kế để phân phối và hiệu quả với những ưu điểm sau:

- Tốc độ đào tạo nhanh và hiệu quả cao
- Sử dụng bộ nhớ ít hơn
- Độ chính xác cao
- Hỗ trợ học song song và GPU,...

2. Thực thi mô hình

Từ tập dữ liệu huấn luyện và kiểm thử đã được chia ở trên tiến hành huấn luyện và đưa ra kết quả mô hình LGBM như Hình 27.

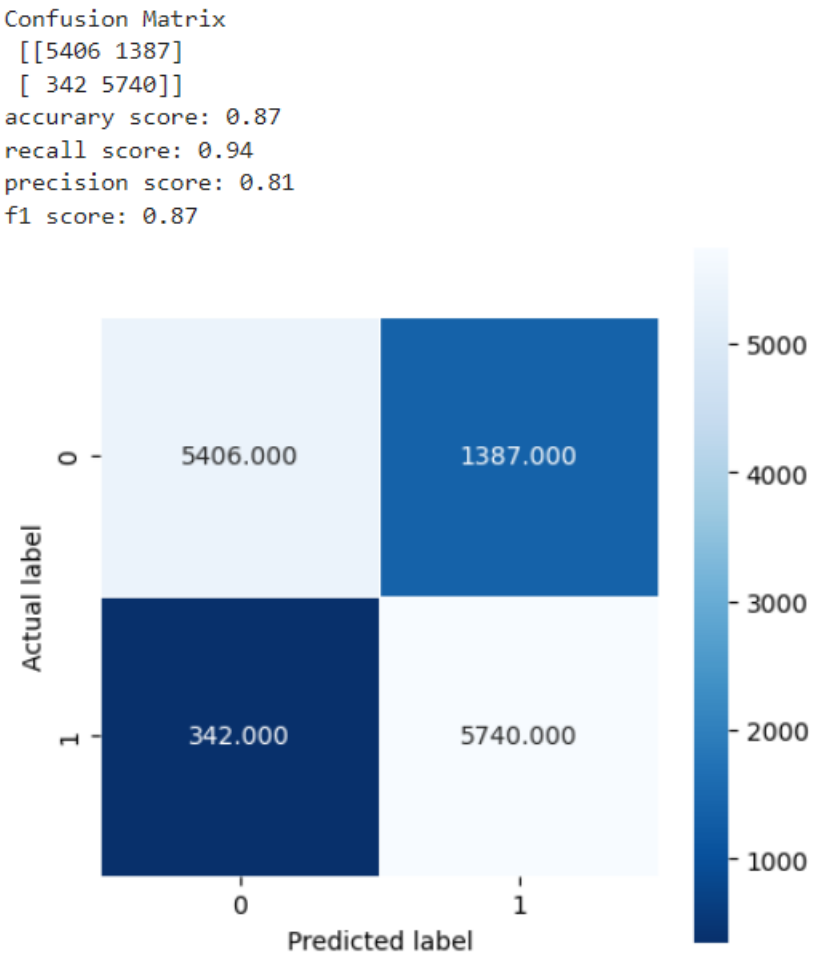
```
import lightgbm as lgb
clf = lgb.LGBMClassifier()
clf.fit(X_train, y_train)
```

```
[LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
[LightGBM] [Info] Number of positive: 24411, number of negative: 27088
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001507
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 132
[LightGBM] [Info] Number of data points in the train set: 51499, number of used features: 4
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.474009 -> initscore=-0.104057
[LightGBM] [Info] Start training from score -0.104057
▾ LGBMClassifier
LGBMClassifier()
```

```
y_pred_test=clf.predict(X_test)
print("Confusion Matrix\n",confusion_matrix(y_test, y_pred_test))
print(f"accurary score: {accuracy_score(y_test,y_pred_test):.2f}")
print(f'recall score: {recall_score(y_test, y_pred_test):.2f}')
print(f'precision score: {precision_score(y_test, y_pred_test):.2f}')
print(f'f1 score: {f1_score(y_test, y_pred_test):.2f}')

cnf_matrix = metrics.confusion_matrix(y_test, y_pred_test)
plt.figure(figsize=(5, 5))
sns.heatmap(cnf_matrix, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

Hình 27: Mã nguồn huấn luyện mô hình LGBM

Kết quả về độ đo Accurary, Recall, precision, F1 và biểu đồ ma trận của mô hình LGBM như Hình 28.



Hình 28: Kết quả sau khi huấn luyện mô hình LGBM

1.3.3 Mô hình hồi quy Logistic

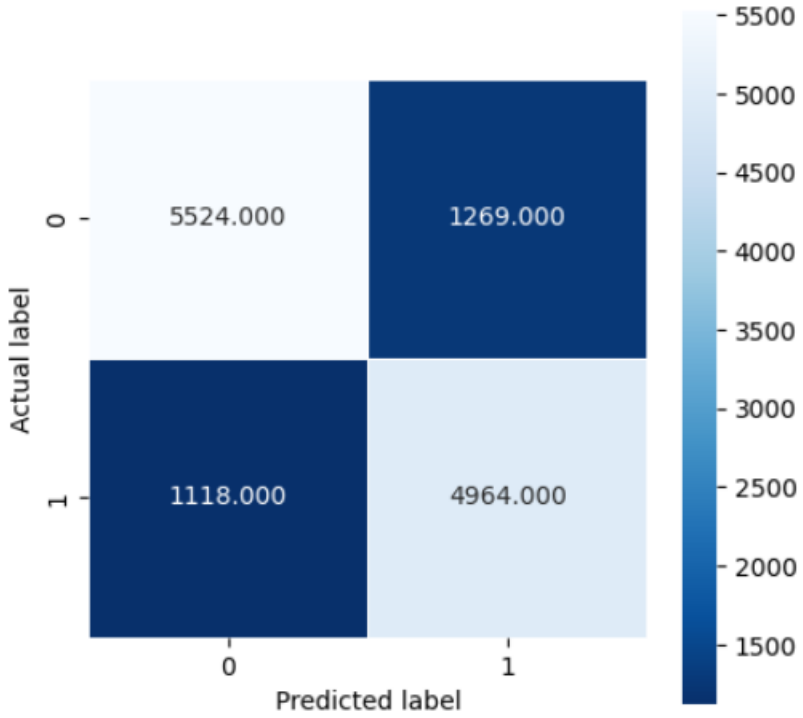
Tạo mô hình và huấn luyện mô hình từ tập huấn luyện và kiểm thử đã tạo từ trước như Hình 29.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Tạo mô hình hồi quy logistic
logistic_model = LogisticRegression()

# Huấn luyện mô hình trên tập huấn luyện
logistic_model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred_test1 = logistic_model.predict(X_test)

# Đánh giá mô hình trên tập kiểm tra
accuracy = accuracy_score(y_test, y_pred_test1)
conf_matrix = confusion_matrix(y_test, y_pred_test1)

# In kết quả
y_pred_test=logistic_model.predict(X_test)
print("Confusion Matrix\n",confusion_matrix(y_test, y_pred_test))
print(f"accurary score: {accuracy_score(y_test,y_pred_test):.2f}")
print(f'recall score: {recall_score(y_test, y_pred_test):.2f}')
print(f'precision score: {precision_score(y_test, y_pred_test):.2f}')
print(f'f1 score: {f1_score(y_test, y_pred_test):.2f}')

cnf_matrix = metrics.confusion_matrix(y_test, y_pred_test)
plt.figure(figsize=(5, 5))
sns.heatmap(cnf_matrix, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

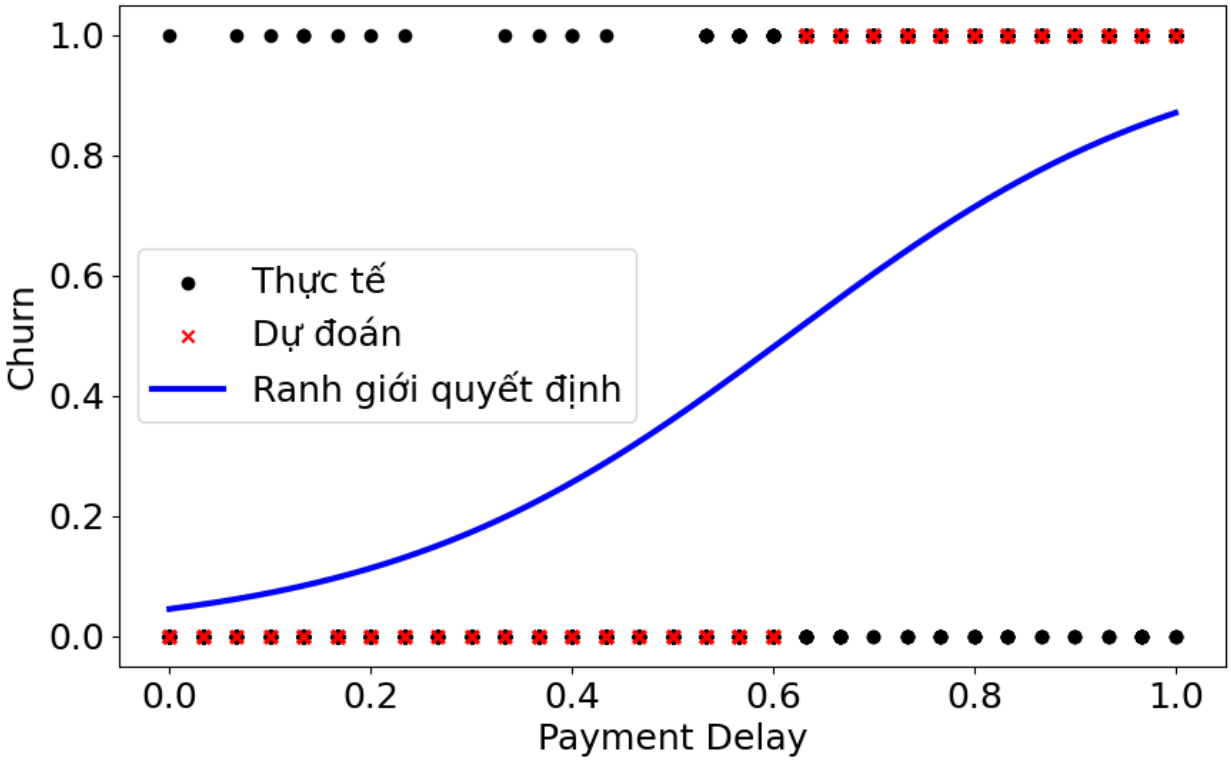
Hình 29: Mã nguồn huấn luyện mô hình hồi quy Logistic

Kết quả về độ đo Accurary, Recall, precision, F1 và biểu đồ ma trận của mô hình hồi quy Logistic như Hình 30.

```
Confusion Matrix
 [[5524 1269]
 [1118 4964]]
accurary score: 0.81
recall score: 0.82
precision score: 0.80
f1 score: 0.81
```



Hình 30: Kết quả sau khi huấn luyện mô hình hồi quy Logistic



Hình 31: Ranh giới quyết định mô hình Logistic

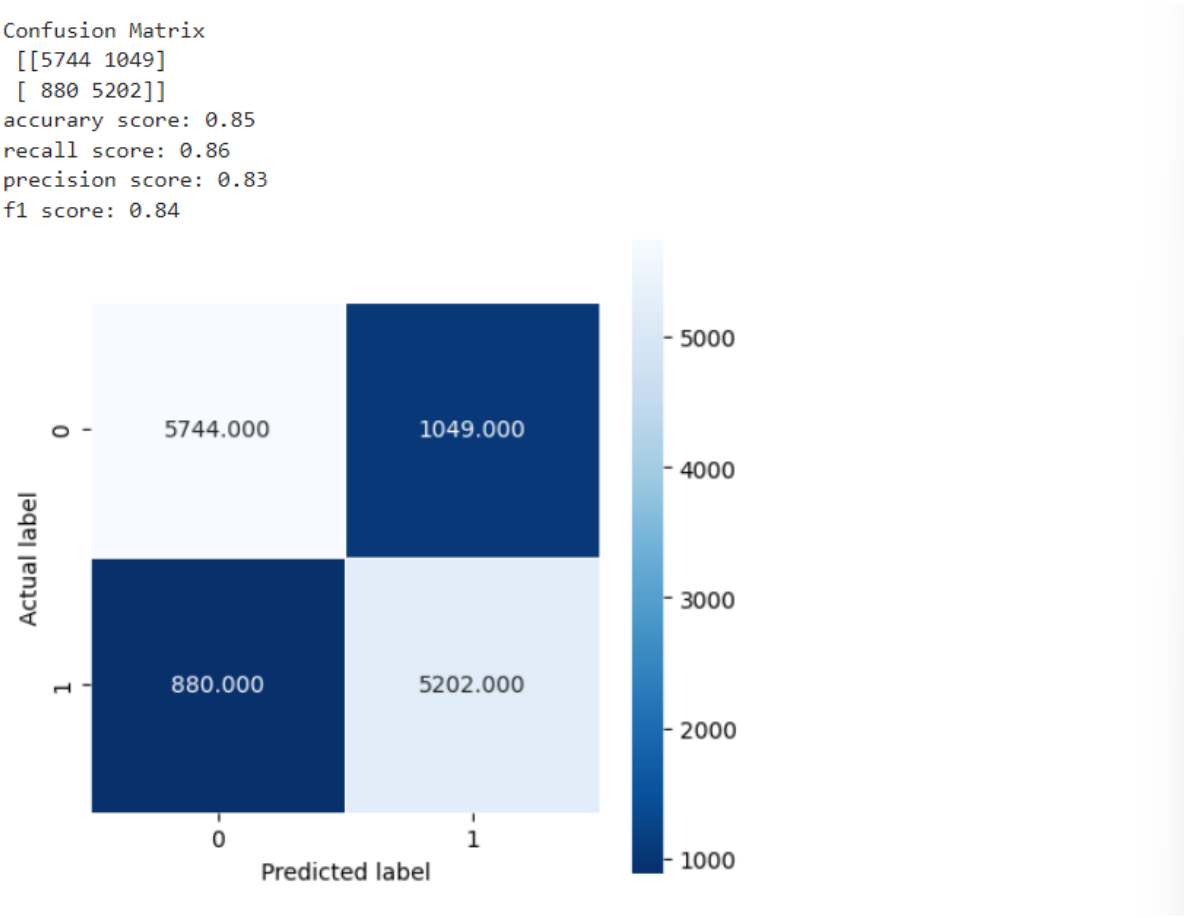
1.3.4 Mô hình K-Nearest Neighbors

Tạo mô hình và huấn luyện mô hình từ tập huấn luyện và kiểm thử đã tạo từ trước như **Error! Reference source not found.**.

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Tạo mô hình KNN với k=3 (số lượng láng giềng cận kề)
knn_model = KNeighborsClassifier(n_neighbors=3)

# Huấn luyện mô hình trên tập huấn luyện
knn_model.fit(X_train, y_train)

# Dự đoán trên tập kiểm tra
y_pred_test = knn_model.predict(X_test)

# Đánh giá mô hình trên tập kiểm tra
accuracy = accuracy_score(y_test, y_pred_test)
conf_matrix = confusion_matrix(y_test, y_pred_test)
classification_rep = classification_report(y_test, y_pred_test)

# In kết quả
y_pred_test=knn_model.predict(X_test)
print("Confusion Matrix\n",confusion_matrix(y_test, y_pred_test))
print(f"accurary score: {accuracy_score(y_test,y_pred_test):.2f}")
print(f'recall score: {recall_score(y_test, y_pred_test):.2f}')
print(f'precision score: {precision_score(y_test, y_pred_test):.2f}')
print(f'f1 score: {f1_score(y_test, y_pred_test):.2f}')

cnf_matrix = metrics.confusion_matrix(y_test, y_pred_test)
plt.figure(figsize=(5, 5))
sns.heatmap(cnf_matrix, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

Hình 32: Mã nguồn huấn luyện mô hình KNN

Kết quả về độ đo Accurary, Recall, precision, F1 và biểu đồ ma trận của mô hình KNN như Hình 33.



Hình 33: Kết quả huấn luyện mô hình KNN

1.3.5 Mô hình Supoort Vecto Machine

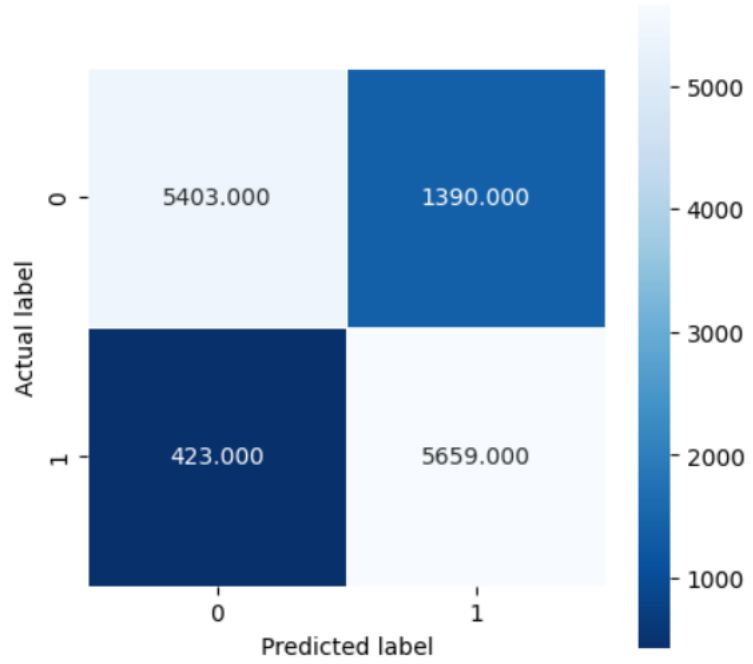
Tạo mô hình và huấn luyện mô hình từ tập huấn luyện và kiểm thử đã tạo từ trước như Hình 34.

```
classifier=SVC(kernel='rbf',C=10)
classifier.fit(X_train,y_train)

# Huấn luyện mô hình trên tập huấn luyện
y_pred_test=classifier.predict(X_test)

# Đánh giá mô hình trên tập kiểm tra
accuracy = accuracy_score(y_test, y_pred_test)
conf_matrix = confusion_matrix(y_test, y_pred_test)
classification_rep = classification_report(y_test, y_pred_test)

# In kết quả
y_pred_test=classifier.predict(X_test)
print("Confusion Matrix\n",confusion_matrix(y_test, y_pred_test))
print(f"accurary score: {accuracy_score(y_test,y_pred_test):.2f}")
print(f'recall score: {recall_score(y_test, y_pred_test):.2f}')
print(f'precision score: {precision_score(y_test, y_pred_test):.2f}')
print(f'f1 score: {f1_score(y_test, y_pred_test):.2f}')
cnf_matrix = metrics.confusion_matrix(y_test, y_pred_test)
plt.figure(figsize=(5, 5))
sns.heatmap(cnf_matrix, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

plt.show()
```

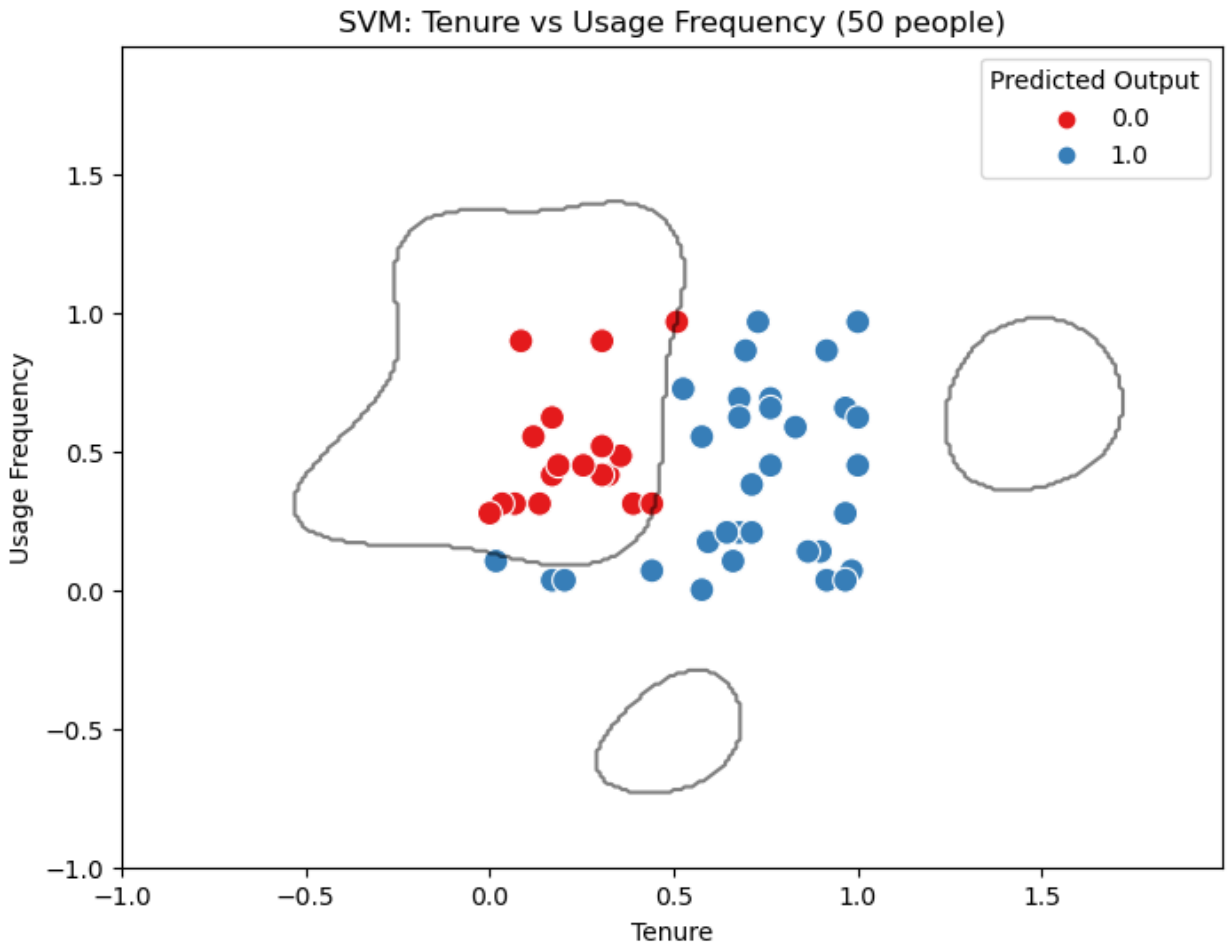
Hình 34: Mã nguồn huấn luyện mô hình SVM

Kết quả về độ đo Accurary, Recall, precision, F1 và biểu đồ ma trận của mô hình KNN như Hình 35.

```
Confusion Matrix
 [[5403 1390]
 [ 423 5659]]
accurary score: 0.86
recall score: 0.93
precision score: 0.80
f1 score: 0.86
```



Hình 35: Kết quả huấn luyện mô hình SVM



Hình 36: Trực quan phân loại mô hình SVM

1.3 Thảo luận, phân tích, đánh giá và kết luận về kết quả nhận được sau khi tích dữ liệu

	Accurary	Recall Score	Precision	F1 Score	Elapsed time (s)
LGBM	0.87	0.94	0.81	0.87	0.08
SVM	0.86	0.93	0.80	0.86	30.87
KNN	0.85	0.86	0.83	0.84	0.29
Logistic	0.81	0.82	0.80	0.81	0.003

1. Mô hình LGBM:
    - Đạt hiệu suất tốt và độ nhớ lại (Recall) cao (tỷ lệ bỏ sót các điểm thực sự là Positive là thấp).
    - Precision khá cao (Khả năng dự đoán các điểm Positive thực sự là Positive là khá cao) .

- Thời gian thực hiện rất nhanh, chỉ mất 0.08 giây.

2. Mô hình SVM:
    - SVM với kernel RBF cũng cho kết quả tốt với độ nhớ lại cao.
    - Precision giảm so với LGBM, nhưng độ chính xác vẫn khá cao.
    - Thời gian thực hiện lớn hơn nhiều so với LGBM.
3. Mô hình KNN:
    - Kết quả độ chính xác và độ nhớ lại ổn định, không quá cao nhưng cũng không quá thấp.
    - Precision và F1 Score đều khá tốt.
    - Thời gian thực hiện cũng khá nhanh.
4. Mô hình hồi quy Logistic:
    - Logistic Regression cung cấp kết quả khá tốt, nhưng thấp hơn so với các mô hình trước.
    - Độ nhớ lại và độ chính xác đều ổn định, nhưng không cao bằng LGBM và SVM.
    - Thời gian thực hiện rất nhanh, chỉ mất 0.003 giây.

Tổng kết:

- LGBM và SVM (kernel RBF) đều có hiệu suất tốt, với SVM có độ nhớ lại cao hơn nhưng cần thời gian tính toán lâu hơn.
- KNN cũng đạt được kết quả tốt, với sự cân bằng giữa độ chính xác và độ nhớ lại.
- Logistic cung cấp kết quả không bằng các mô hình khác nhưng thời gian thực hiện rất nhanh.

PHẦN 2: HỌC KHÔNG GIÁM SÁT (UNSUPERVISED LEARNING)

2.1 Giới thiệu mục đích bài toán

2.1.1 Giới thiệu dữ liệu

Tập dữ liệu sử dụng trong phần học không giám sát này liên quan đến chi tiêu các mặt hàng. Dưới đây

* ID: Định danh duy nhất của khách hàng
* Year_Birth: Năm sinh của khách hàng
* Education: Trình độ học vấn của khách hàng
* Marital_Status: Tình trạng hôn nhân của khách hàng
* Income: Thu nhập hàng năm của hộ gia đình khách hàng
* Kidhome: Số lượng trẻ em trong hộ gia đình của khách hàng
* Teenhome: Số lượng thanh thiếu niên trong hộ gia đình của khách hàng
* Dt_Customer: Ngày đăng ký của khách hàng với công ty
* Recency: Số ngày kể từ lần mua cuối cùng của khách hàng
* Complain: 1 nếu khách hàng phàn nàn trong vòng 2 năm qua, 0 nếu ngược lại
* MntWines: Số tiền chi tiêu cho rượu trong 2 năm qua
* MntFruits: Số tiền chi tiêu cho trái cây trong 2 năm qua
* MntMeatProducts: Số tiền chi tiêu cho thịt trong 2 năm qua
* MntFishProducts: Số tiền chi tiêu cho cá trong 2 năm qua
* MntSweetProducts: Số tiền chi tiêu cho đồ ngọt trong 2 năm qua
* MntGoldProds: Số tiền chi tiêu cho vàng trong 2 năm qua
* NumDealsPurchases: Số lượng mua hàng với giảm giá
* AcceptedCmp1: 1 nếu khách hàng chấp nhận ưu đãi trong chiến dịch 1, 0 nếu ngược lại
* AcceptedCmp2: 1 nếu khách hàng chấp nhận ưu đãi trong chiến dịch 2, 0 nếu ngược lại
* AcceptedCmp3: 1 nếu khách hàng chấp nhận ưu đãi trong chiến dịch 3, 0 nếu ngược lại
* AcceptedCmp4: 1 nếu khách hàng chấp nhận ưu đãi trong chiến dịch 4, 0 nếu ngược lại
* AcceptedCmp5: 1 nếu khách hàng chấp nhận ưu đãi trong chiến dịch 5, 0 nếu ngược lại
* Response: 1 nếu khách hàng chấp nhận ưu đãi trong chiến dịch cuối cùng, 0 nếu ngược lại
* NumWebPurchases: Số lượng mua hàng thông qua trang web của công ty
* NumCatalogPurchases: Số lượng mua hàng sử dụng catalog
* NumStorePurchases: Số lượng mua hàng trực tiếp tại cửa hàng
* NumWebVisitsMonth: Số lượng lượt truy cập trang web của công ty trong tháng qua

2.2 Tiền xử lí dữ liệu

Dữ liệu ban đầu gồm loại dữ liệu số và dữ liệu phân loại không nhất quán, để phục vụ cho việc trực quan hóa dữ liệu và thực hiện bài toán phân cụm việc tiền xử lí dữ liệu đóng vai trò rất quan trọng.

2.2.1 Làm sạch dữ liệu

1. Kiểm tra giá trị trùng lặp

```
# kiểm tra giá trị trùng lặp
df.duplicated().sum()
```

```
0
```

Hình 37: Kiểm tra giá trị trùng lặp

2. Xử lí giá trị thiếu

```
# Tìm missing values
df.isna().sum()
```

```
ID_                  0
Year_Birth           0
Education            0
Marital_Status       0
Income              24
```

Hình 38: Kiểm tra giá trị thiếu

```
# xoa cac hang chua gia tri thieu
df = df.dropna()
df.isna().sum()
```

```
ID_                  0
Year_Birth           0
Education            0
Marital_Status       0
Income               0
Kidhome              0
```

Hình 39: Xử lí giá trị thiếu

3. Loại bỏ cột không cần thiết

Vì cột "Z_CostContact" và "Z_Revenue" có 1 giá trị duy nhất, có thể loại bỏ hai cột này đi.

```
df = df.drop(['Z_CostContact','Z_Revenue'],axis=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2216 entries, 0 to 2239
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID_                  2216 non-null   int64  
 1   Year_Birth           2216 non-null   int64  
 2   Education            2216 non-null   object 
 3   Marital_Status       2216 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2216 non-null   int64  
 6   Teenhome             2216 non-null   int64  
 7   Dt_Customer          2216 non-null   object 
 8   Recency              2216 non-null   int64  
 9   MntWines             2216 non-null   int64  
 10  MntFruits            2216 non-null   int64  
 11  MntMeatProducts      2216 non-null   int64  
 12  MntFishProducts      2216 non-null   int64  
 13  MntSweetProducts     2216 non-null   int64  
 14  MntGoldProds         2216 non-null   int64  
 15  NumDealsPurchases    2216 non-null   int64  
 16  NumWebPurchases      2216 non-null   int64  
 17  NumCatalogPurchases  2216 non-null   int64  
 18  NumStorePurchases    2216 non-null   int64  
 19  NumWebVisitsMonth    2216 non-null   int64  
 20  AcceptedCmp3         2216 non-null   int64  
 21  AcceptedCmp4         2216 non-null   int64  
 22  AcceptedCmp5         2216 non-null   int64  
 23  AcceptedCmp1         2216 non-null   int64  
 24  AcceptedCmp2         2216 non-null   int64  
 25  Complain             2216 non-null   int64  
 26  Response             2216 non-null   int64  
```

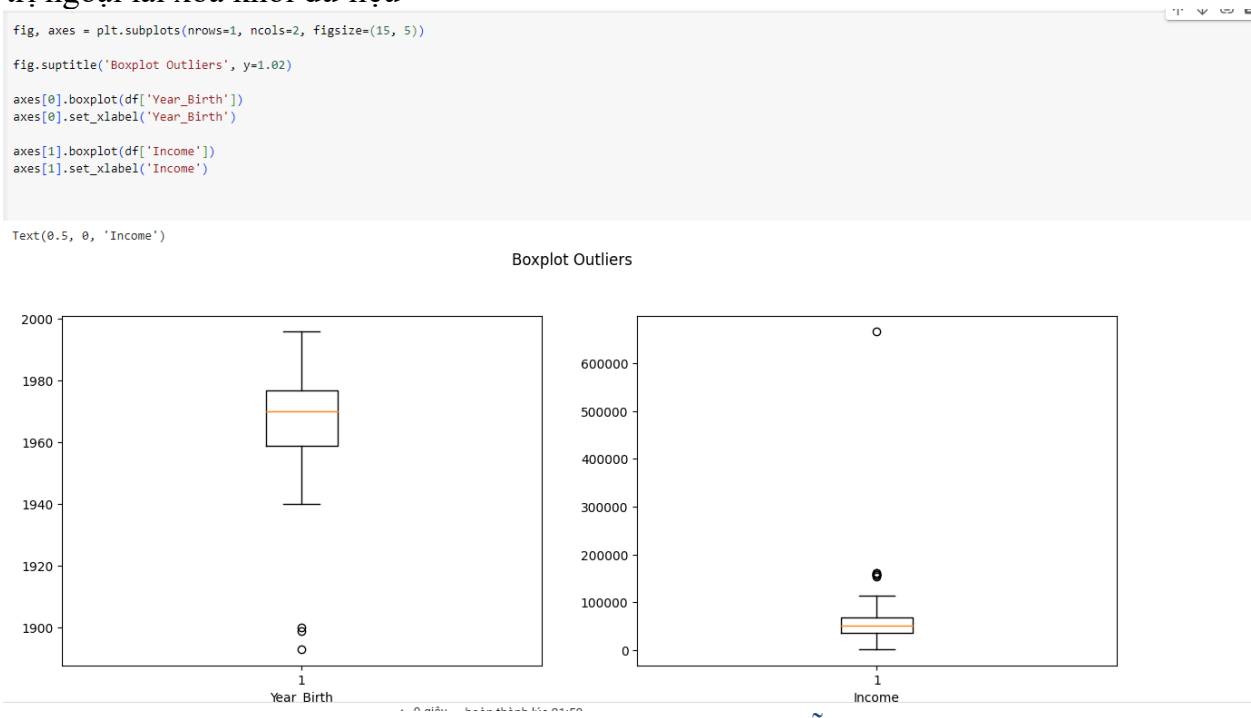
Hình 40: Loại bỏ cột không cần thiết

4. Xử lí giá trị nhiễu

Xử lí giá trị nhiễu bằng cách sử dụng biểu đồ hộp, từ đó có thể nhận ra giá trị ngoại lai xóa khỏi dữ liệu

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15, 5))

fig.suptitle('Boxplot Outliers', y=1.02)

axes[0].boxplot(df['Year_Birth'])
axes[0].set_xlabel('Year_Birth')

axes[1].boxplot(df['Income'])
axes[1].set_xlabel('Income')
```

```
Text(0.5, 0, 'Income')
```



Hình 41: Xử lí giá trị nhiễu

2.2.2 Tích hợp dữ liệu

Dựa vào tính năng các cột để có thể tích hợp các cột lại với nhau thành 1 cột:

```
[23] # tạo cột mới là Age từ cột Year_Birth, rồi xóa cột đó
     df['Age'] = 2023 - df['Year_Birth']
     df = df.drop('Year_Birth',axis=1)
     # trạng thái hôn nhân
     df['Living_With'] = df["Marital_Status"].replace({"Married":2, "Together":2, "Absurd":1, "Widow":1, "YOLO":1, "Divorced":1,'Single':1,'Alone':1})
     df = df.drop('Marital_Status',axis=1)
     df['Children'] = df['Kidhome'] + df['Teenhome']
     df = df.drop(['Kidhome','Teenhome'],axis=1)
     df['Family_Size'] = df['Living_With'] + df['Children']

     df['Total_Spent'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntSweetProducts'] + df['MntGoldProds'] + df['MntFishProducts']
     df['Total_Accept'] = df['AcceptedCmp1'] + df['AcceptedCmp2'] + df['AcceptedCmp3'] + df['AcceptedCmp4'] + df['AcceptedCmp5'] + df['Response']
```

Hình 42: Tích hợp dữ liệu

2.2.3 Chuyển đổi dữ liệu

1. Chuyển hóa dữ liệu

Chuyển hóa dữ liệu dạng phân loại sang dạng số như dưới đây:

```
4] #  chuyển dạng dữ liệu phân loại sang dạng số
   df['Education'] = df['Education'].replace({"Basic": 0, "2n Cycle":1, "Graduation": 2, "Master": 3, "PhD": 4})
```

Hình 43: Chuyển hóa dữ liệu

2. Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu là một bước quan trọng trong quá trình tiền xử lý dữ liệu, nhằm chuyển các đặc trưng định lượng (number) về cùng một thang đo chung, giúp cho việc biểu diễn dữ liệu dễ dàng và các mô hình phân tích nhất là các mô hình học máy (machine learning) hoạt động hiệu quả hơn.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
colums_scale=['Income','Recency','MntWines','MntFruits','MntMeatProducts',
              'MntFishProducts','MntSweetProducts','MntGoldProds','NumDealsPurchases','NumWebPurchases','NumCatalogPurchases','NumStorePurchases','NumWebVisitsMonth','Age']
df[colums_scale] = MinMaxScaler().fit_transform(df[colums_scale])
```

Hình 44: Chuẩn hóa dữ liệu

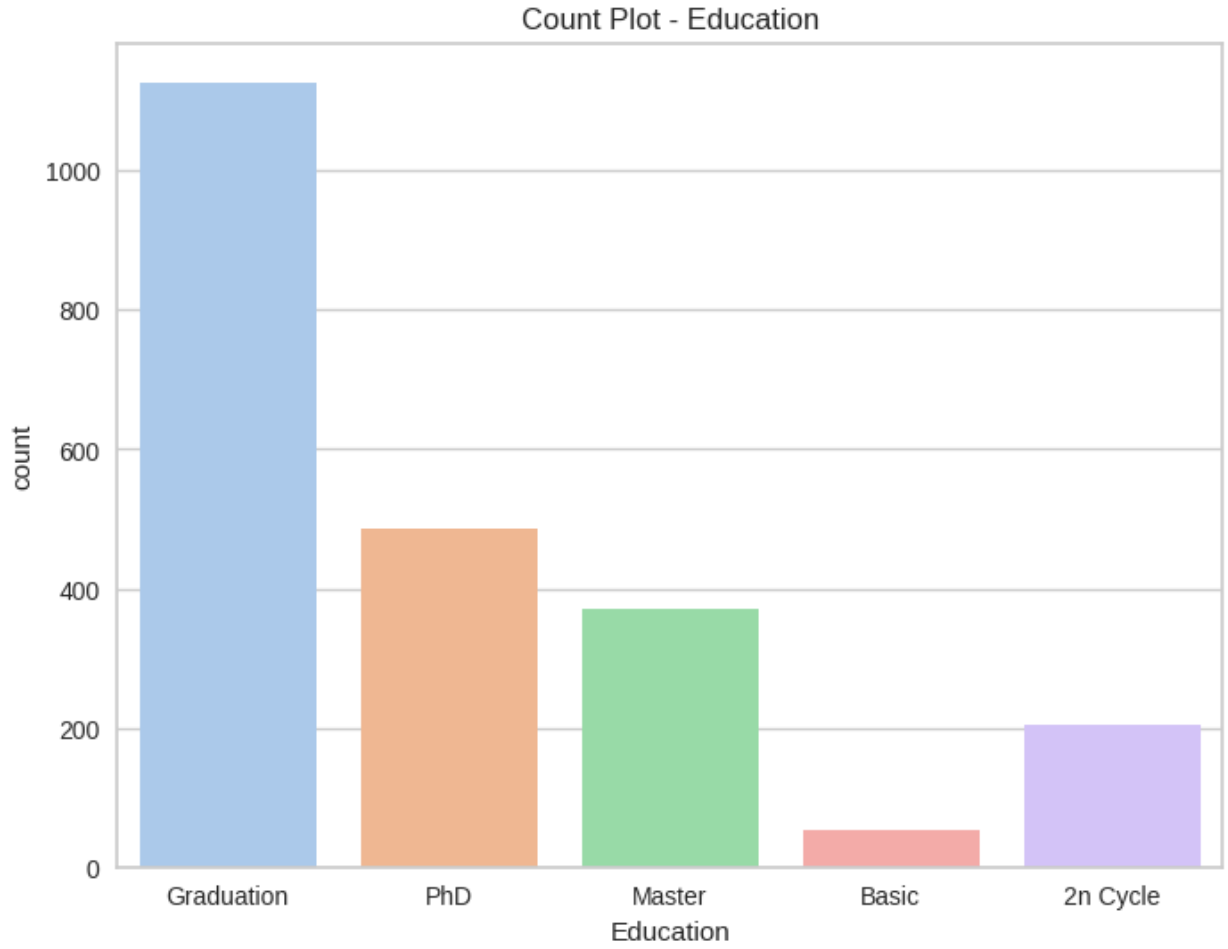
Sau khi kết thúc quá trình tiền xử lí dữ liệu, kết quả như dưới đây

Education	Income	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	NumDealsPurchases	...	AcceptedCmp1	AcceptedCmp2	Complain	Response	Age
2	0.084832	0.585859	0.425318	0.442211	0.316522	0.664093	0.335878	0.274143	0.200000	...	0	0	0	1	0.378641
2	0.067095	0.383838	0.007368	0.005025	0.003478	0.007722	0.003817	0.018692	0.133333	...	0	0	0	0	0.407767
2	0.105097	0.262626	0.285332	0.246231	0.073623	0.428571	0.080153	0.130841	0.066667	...	0	0	0	0	0.300971
2	0.037471	0.262626	0.007368	0.020101	0.011594	0.038610	0.011450	0.015576	0.133333	...	0	0	0	0	0.116505
4	0.085065	0.949495	0.115874	0.216080	0.068406	0.177606	0.103053	0.046729	0.333333	...	0	0	0	0	0.145631
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2	0.089472	0.464646	0.474883	0.216080	0.105507	0.162162	0.450382	0.769470	0.133333	...	0	0	0	0	0.281553
4	0.093669	0.565657	0.271936	0.000000	0.017391	0.000000	0.000000	0.024922	0.466667	...	1	0	0	0	0.485437
2	0.083092	0.919192	0.608171	0.241206	0.125797	0.123552	0.045802	0.074766	0.066667	...	0	0	0	0	0.145631
3	0.101536	0.080808	0.286671	0.150754	0.124058	0.308880	0.114504	0.190031	0.133333	...	0	0	0	0	0.388350
4	0.076908	0.404040	0.056263	0.015075	0.035362	0.007722	0.003817	0.065421	0.200000	...	0	0	0	1	0.407767

Hình 45: Kết quả tiền xử lí dữ liệu

2.3 Trực quan hóa dữ liệu

2.3.1 Biểu đồ cột



Hình 46: Biểu đồ cột tình trạng học vấn

Từ kết quả biểu đồ cho thấy trình độ học vấn đạt bằng tốt nghiệp đại học nhiều hơn bất kì trình độ học vấn nào, điều này cũng dễ hiểu khi các công việc hiện tại có bằng cấp vẫn là yếu tố đáng chú ý.

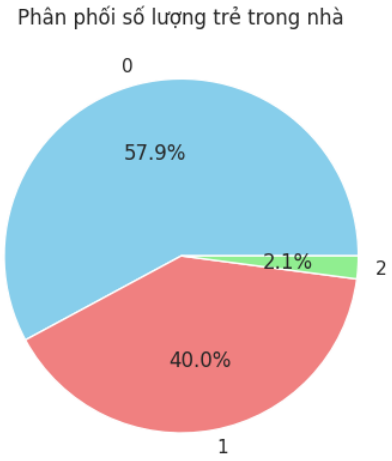
2.3.2 Biểu đồ đường

Hình 47: Biểu đồ đường dao động ngày cuối cùng mua với thu nhập

Kết quả biểu đồ đường cho thấy giao động thu nhập của các người có số ngày kể từ lần mua cuối cùng là không đồng đều.

2.3.3 Biểu đồ tròn

```
# Tính số lượng trẻ em có trong gia đình
kidhome_counts = df['Kidhome'].value_counts()

# Vẽ biểu đồ tròn
plt.pie(kidhome_counts, labels=kidhome_counts.index, autopct='%1.1f%%', colors = ['skyblue', 'lightcoral', 'lightgreen'])
plt.title('Phân phối số lượng trẻ trong nhà')

# Hiển thị biểu đồ
plt.show()
```

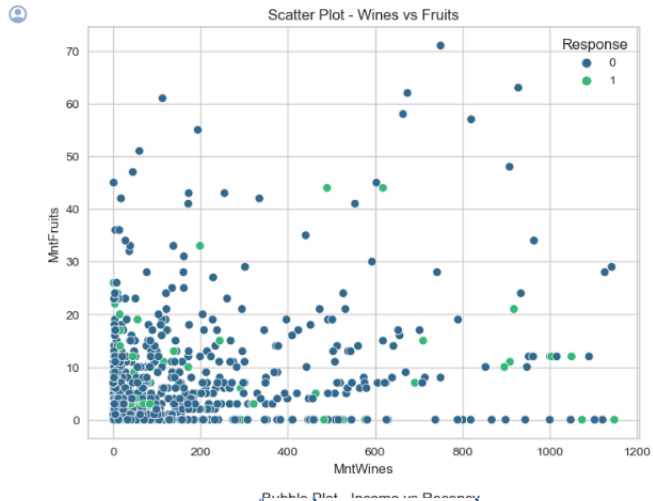


Hình 48: Biểu đồ tròn phân phối số lượng trẻ trong gia đình

Từ biểu đồ tròn phân phối số lượng trẻ em có trong gia đình, cho thấy số lượng trẻ em không có trong gia đình chiếm gần 58%, trong khi 2 yếu tố còn lại chỉ chiếm hơn 42%. Từ đó cho thấy đây cũng là một yếu tố để phân cụm chi tiêu khách hàng.

2.3.4 Biểu đồ phân tán

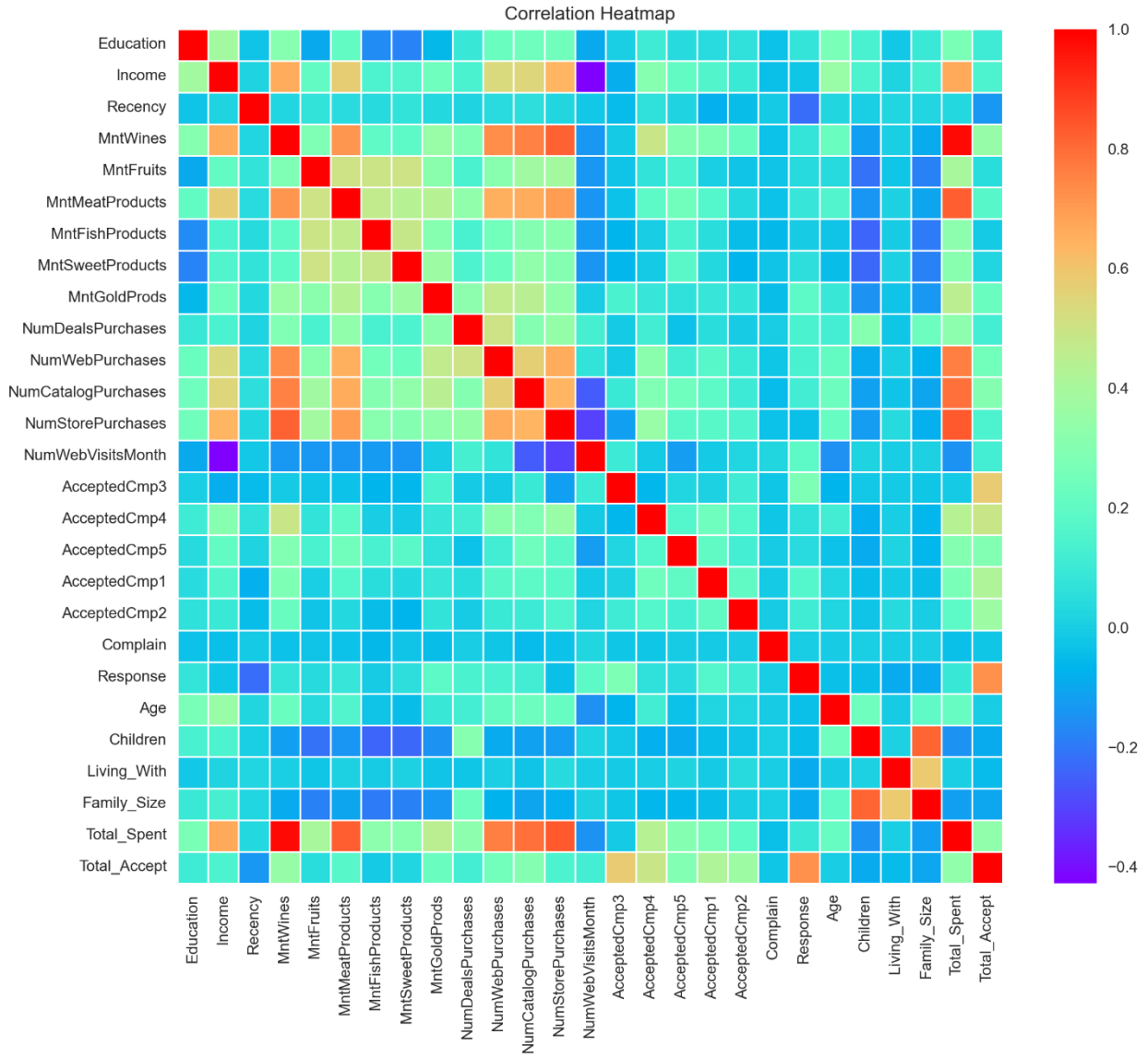
```
# Biểu đồ điểm
plt.figure(figsize=(8, 6))
sns.scatterplot(x='MntWines', y='MntFruits', data=df, hue='Response', palette='viridis')
plt.title('Scatter Plot - Wines vs Fruits')
plt.show()
```



Hình 49: Biểu đồ phân tán tiền mua rượu và tiền mua trái cây

Biểu đồ phân tán tiền mua rượu và trái cái cây của mỗi người và được nhận ưu đãi cho chiến dịch lần cuối hay không.

2.3.5 Biểu đồ nhiệt



Hình 50: Biểu đồ nhiệt

Biểu đồ nhiệt thể hiện tổng quát hệ số tương quan các cột dữ liệu trong datasets với nhau.

2.4 Ứng dụng các mô hình vào bài toán

2.4.1 Thuật toán PCA (Principal component analysis)

PCA là viết tắt của "Principal Component Analysis," một phương pháp trong thống kê và máy học được sử dụng để giảm số chiều của dữ liệu trong không gian đặc trưng. Mục

tiêu của PCA là giảm sự phức tạp của dữ liệu bằng cách chuyển đổi các biến tương quan thành các biến không tương quan mới, gọi là các thành phần chính (principal components). Các thành phần chính được sắp xế theo độ giảm dần của phương sai, với ý nghĩa là thành phần đầu tiên giữ lại nhiều thông tin nhất về biến đổi của dữ liệu.

Ở bộ dữ liệu này, chúng tôi giảm chiều PCA về 2 chiều:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(df_scaled)
```

```
PCA
PCA(n_components=2)
```

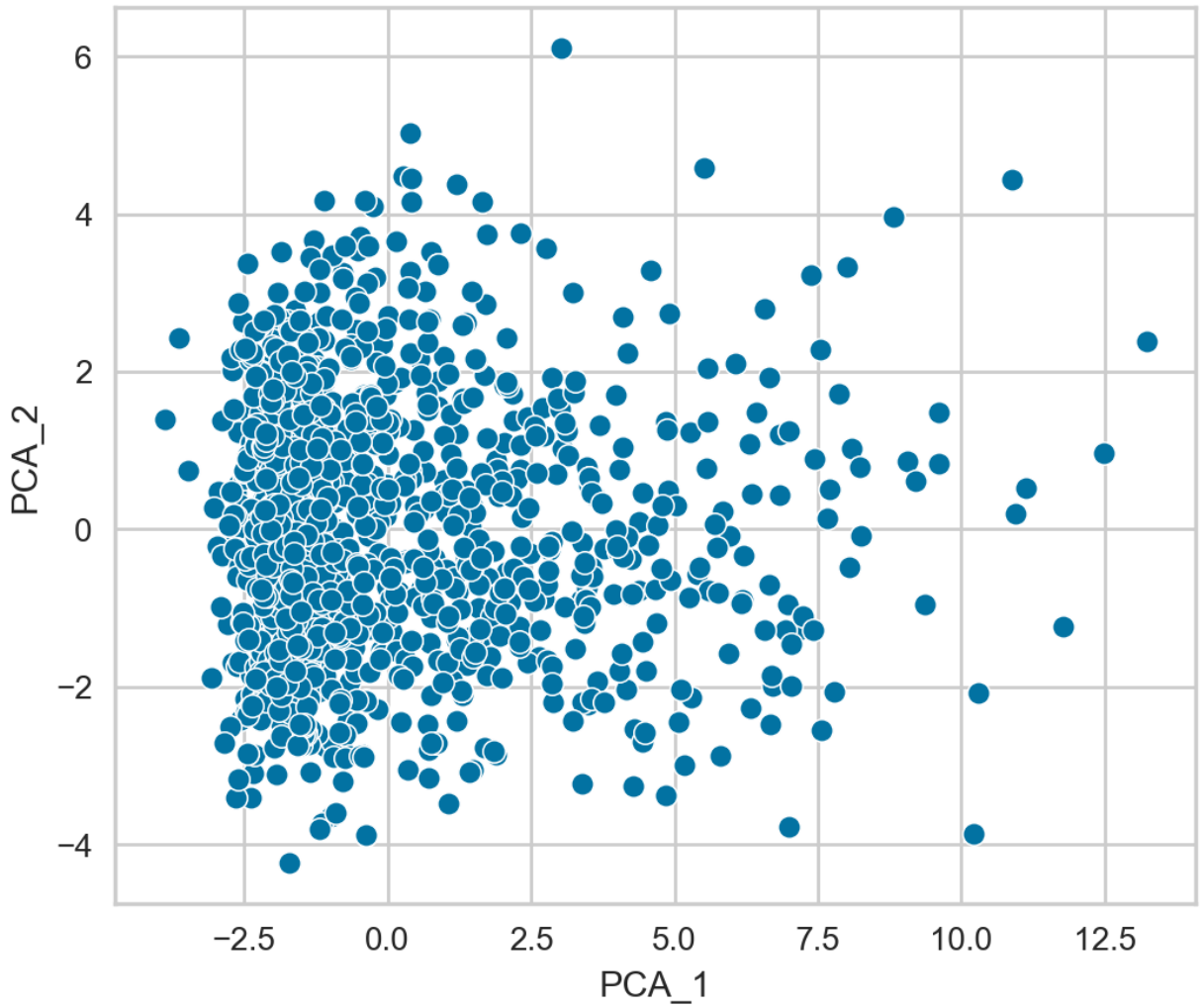
Hình 51: Triển khai thuật toán PCA

Trực quan hóa bằng Biểu đồ điểm:

```
df_reduction = pd.DataFrame(pca.transform(df_scaled), columns=['PCA_1','PCA_2'])
```

```
plt.figure(figsize=(6,5),dpi=200)
sns.scatterplot(x=df_reduction['PCA_1'],y=df_reduction['PCA_2'])
```

Hình 52: Mã nguồn biểu đồ điểm sau khi giảm chiều



Hình 53: Biểu đồ phân tán sau khi áp dụng PCA

Việc giảm chiều này sẽ hỗ trợ trong việc trực quan hóa cho những mô hình học không giám sát dưới đây.

2.4.2 Mô hình Kmeans

K-Means Clustering là một thuật toán trong machine learning và data mining được sử dụng để phân loại một tập dữ liệu thành các nhóm (clusters) dựa trên các đặc trưng của chúng. Phương pháp này thuộc về loại các thuật toán unsupervised learning, nghĩa là không yêu cầu thông tin nhãn trước đó về từng điểm dữ liệu.

Tạo một biến data mới cho việc phân cụm K-Means:

```
df_reduction1 = df_reduction.copy()
df_reduction1
```

	PCA_1	PCA_2
0	-1.556245	-1.176028
1	-1.581689	0.555600
2	4.905551	2.735371
3	-0.345759	-0.088841
4	-1.402452	-0.515081
...	...	...
1205	6.646240	1.928438
1206	-2.155633	2.647937
1207	-2.134485	0.242618
1208	8.806652	3.967336
1209	0.747920	-2.719591

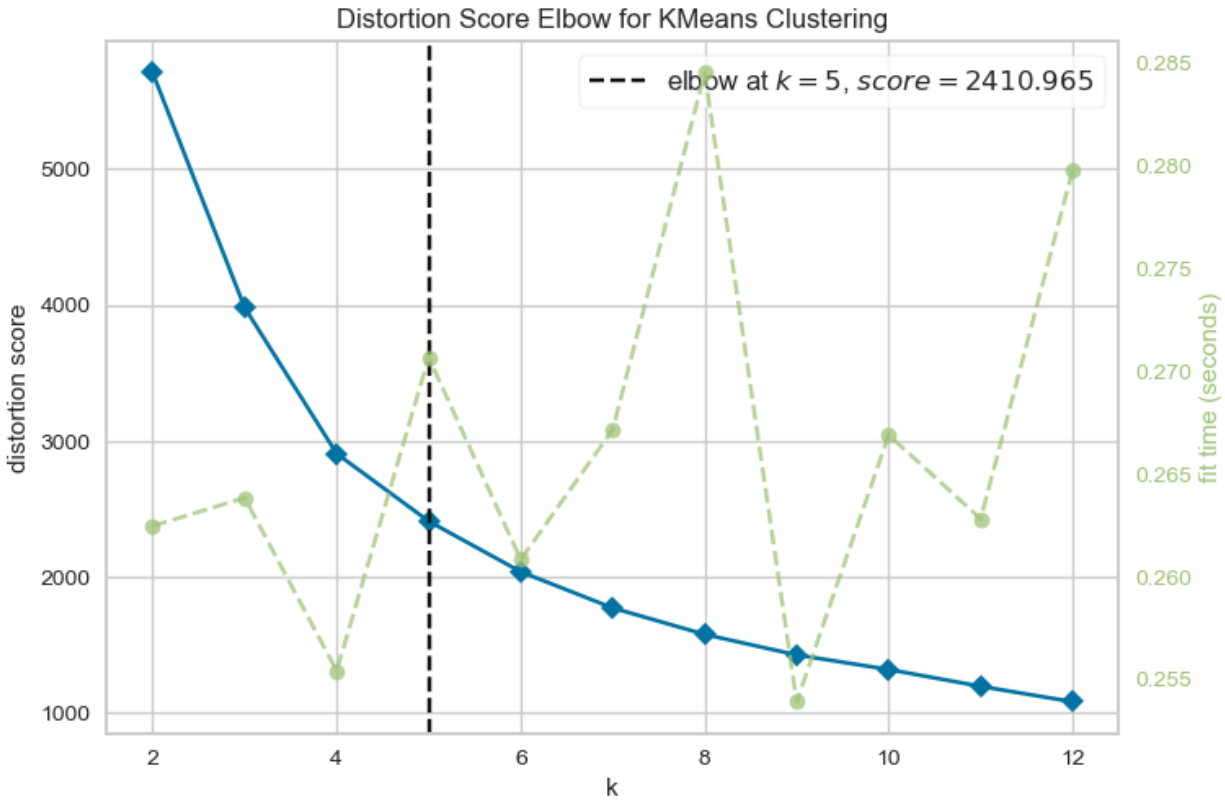
1210 rows × 2 columns

Hình 54: Chuẩn bị dữ liệu cho K-Means

Vẽ đường Elbow để xác định số K hợp lí:

```
elbow = KElbowVisualizer(estimator=KMeans(), k=12, metric='distortion')
elbow.fit(df_reduction1)
elbow.show()
```

Hình 55: Mã nguồn tìm hệ số k phù hợp cho model K-Means



Hình 56: Kết quả tìm hệ số K cho mô hình K-means

Từ kết quả ta thấy số K hợp lí là 5

Khởi tạo mô hình K-Means như dưới đây.

```
KMeans_model = KMeans(n_clusters=5,random_state=101)
y_pred = KMeans_model.fit_predict(df_reduction1)

#Add cluster to reduction dataset
df_reduction1['Cluster'] = y_pred

#Add cluster to original dataset
df['Cluster'] = y_pred
```

Hình 57: Mã nguồn huấn luyện mô hình K-Means

Kết quả sau thực hiện mô hình K-means phân cụm như dưới đây.

```
df_reduction1
```

	PCA_1	PCA_2	Cluster
0	-1.556245	-1.176028	3
1	-1.581689	0.555600	0
2	4.905551	2.735371	2
3	-0.345759	-0.088841	0
4	-1.402452	-0.515081	3
...	...	...	...
1205	6.646240	1.928438	2
1206	-2.155633	2.647937	0
1207	-2.134485	0.242618	0
1208	8.806652	3.967336	2
1209	0.747920	-2.719591	3

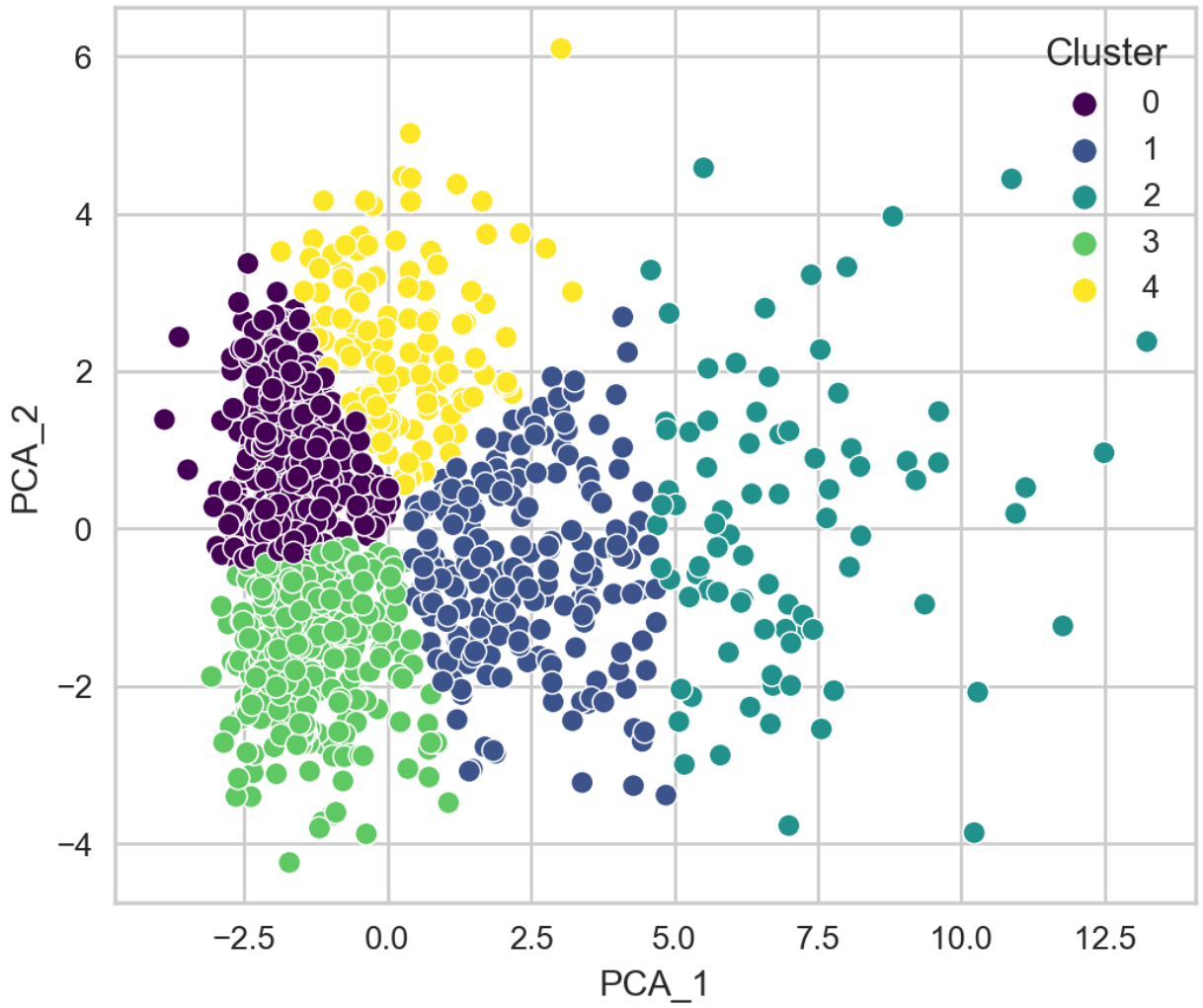
1210 rows × 3 columns

Hình 58: Kết quả phân cụm bằng K-means

Trực quan hóa dữ liệu sau khi phân cụm bằng K-Means như dưới đây.

```
plt.figure(figsize=(6,5),dpi=200)
sns.scatterplot(x=df_reduction1['PCA_1'],y=df_reduction1['PCA_2'],
                hue=df_reduction1['Cluster'],  palette='viridis', legend='full')
```

Hình 59: Mã nguồn trực quan hóa phân cụm sau khi dùng K-means



Hình 60: Biểu đồ phân cụm bằng K-means

2.4.3 Mô hình DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) là một thuật toán phân cụm (clustering) dựa trên mật độ trong không gian đặc trưng. Điều này có nghĩa là DBSCAN tập trung vào việc phát hiện các vùng có mật độ cao của điểm dữ liệu và tự động phân loại chúng thành các cụm (clusters). DBSCAN cũng có khả năng xác định các điểm nhiễu, tức là các điểm không thuộc vào bất kỳ cluster nào.

Khởi tạo mô hình DBSCAN như dưới đây:

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

eps = 0.5
min_samples = 15
dbscan = DBSCAN(eps=eps, min_samples=min_samples,  algorithm='auto',)
clusters = dbscan.fit_predict(df_reduction2)
```

Hình 61: Khởi tạo mô hình phân cụm bằng DBSCAN

Kết quả sau khi phân cụm bằng Dbscan như hình ảnh dưới đây:

```
df_reduction2
```

	PCA_1	PCA_2	cluster
0	-1.556245	-1.176028	0
1	-1.581689	0.555600	0
2	4.905551	2.735371	-1
3	-0.345759	-0.088841	0
4	-1.402452	-0.515081	0
...	...	...	...
1205	6.646240	1.928438	-1
1206	-2.155633	2.647937	0
1207	-2.134485	0.242618	0
1208	8.806652	3.967336	-1
1209	0.747920	-2.719591	-1

1210 rows × 3 columns

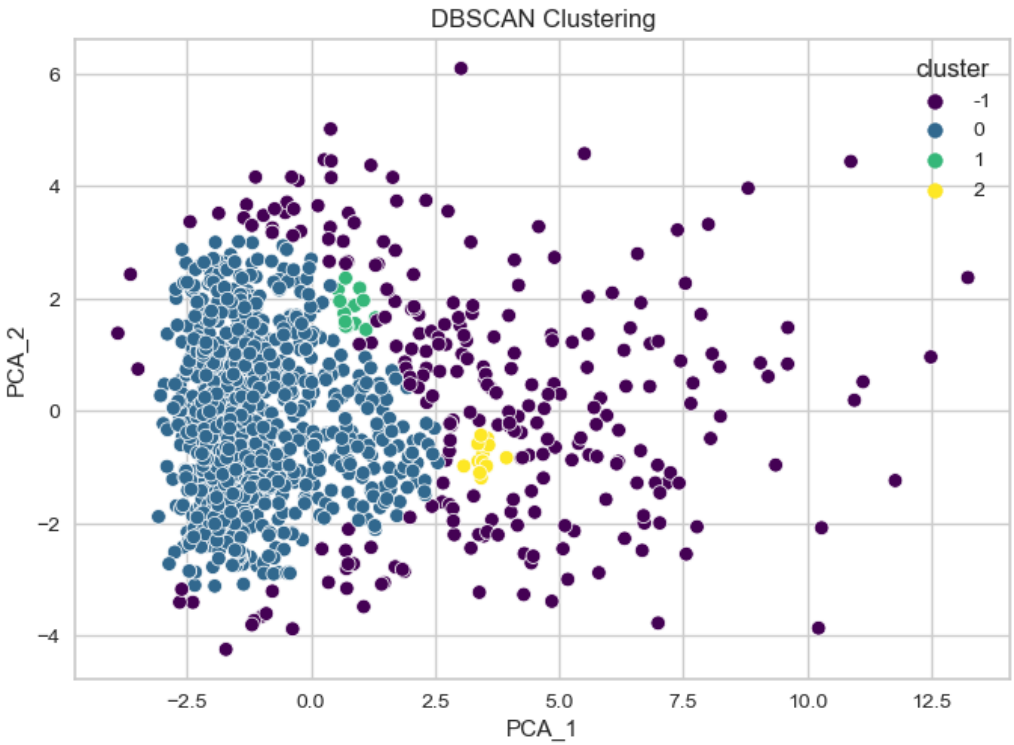
Hình 62: Kết quả phân cụm bằng DBSCAN

Trực quan hóa kết quả phân cụm bằng DBScan như hình dưới đây:

```
import seaborn as sns

# Assuming df_reduction2 is a DataFrame with columns 'PCA_1' and 'PCA_2'
df_reduction2['cluster'] = clusters

# Visualize the results
sns.scatterplot(x=df_reduction2['PCA_1'], y=df_reduction2['PCA_2'],
                hue=df_reduction2['cluster'], palette='viridis', legend='full')

# Set plot labels
plt.title('DBSCAN Clustering')
plt.xlabel('PCA_1')
plt.ylabel('PCA_2')

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

Hình 63: Mã nguồn trực quan hóa phân cụm bằng DBScan

Dưới đây là kết quả phân cụm bằng DBScan bằng biểu đồ trực quan



Hình 64: Kết quả trực quan hóa phân cụm bằng DBScan

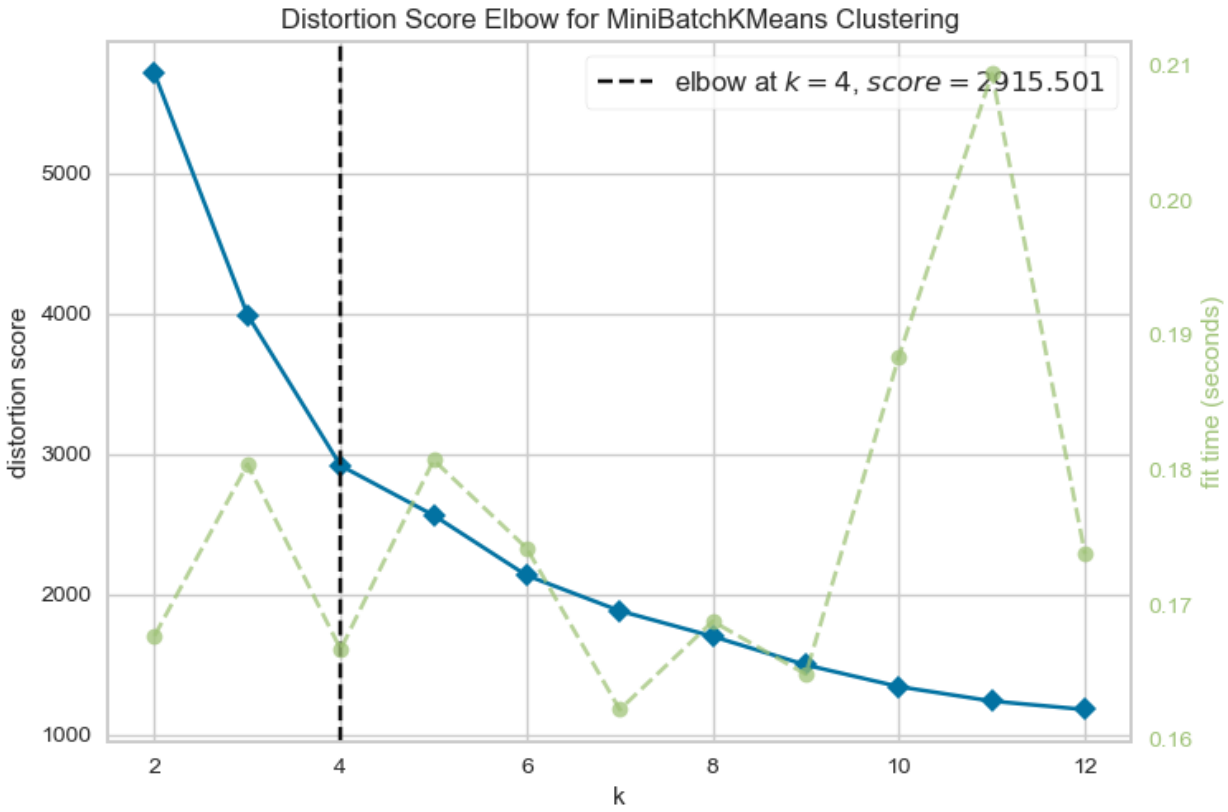
2.4.4 Mô hình MiniBatchKMeans

MiniBatchKMeans là một biến thể của thuật toán K-Means Clustering được thiết kế để xử lý dữ liệu lớn một cách hiệu quả hơn. Trong thuật toán K-Means truyền thống, toàn bộ tập dữ liệu được sử dụng để cập nhật centroids và phân loại các điểm vào các nhóm. Điều này có thể trở nên không hiệu quả khi làm việc với dữ liệu lớn.
MiniBatchKMeans giả định rằng chỉ một phần nhỏ (mini-batch) của dữ liệu được sử dụng để cập nhật centroids ở mỗi bước lặp. Điều này giúp giảm độ phức tạp tính toán và làm tăng tốc độ hội tụ của thuật toán. Bằng cách này, MiniBatchKMeans có thể xử lý các tập dữ liệu lớn mà vẫn giữ được tính chính xác tương đương với K-Means truyền thống.

Vì là 1 biến thể của KMeans Clustering, MiniBatchKMeans cũng cần xác định số k bằng Elbow.

```
elbow = KElbowVisualizer(estimator=MiniBatchKMeans(), k=12, metric='distortion')
elbow.fit(df_reduction_t)
elbow.show()
```

Hình 65: Mã nguồn tìm hệ số K cho MiniBactch Kmeans



Hình 66: Kết quả tìm hệ số K cho MinniBatch Kmeans

Sau khi xác định tham số K tốt nhất cho mình thì tiến hành xây dựng mô hình MiniBatchKMeans:

```
from sklearn.cluster import MiniBatchKMeans
KMeans_model = MiniBatchKMeans(n_clusters=4,random_state=42)
y_pred = KMeans_model.fit_predict(df_reduction_t)
#Add cluster to reduction dataset
df_reduction_t['Cluster'] = y_pred
df['Clusterm'] = y_pred
```

Hình 67: Mã nguồn mô hình MiniBatchKmeans

Kết quả sau phân cụm bằng mô hình MiniBatch Kmeans, như dưới đây

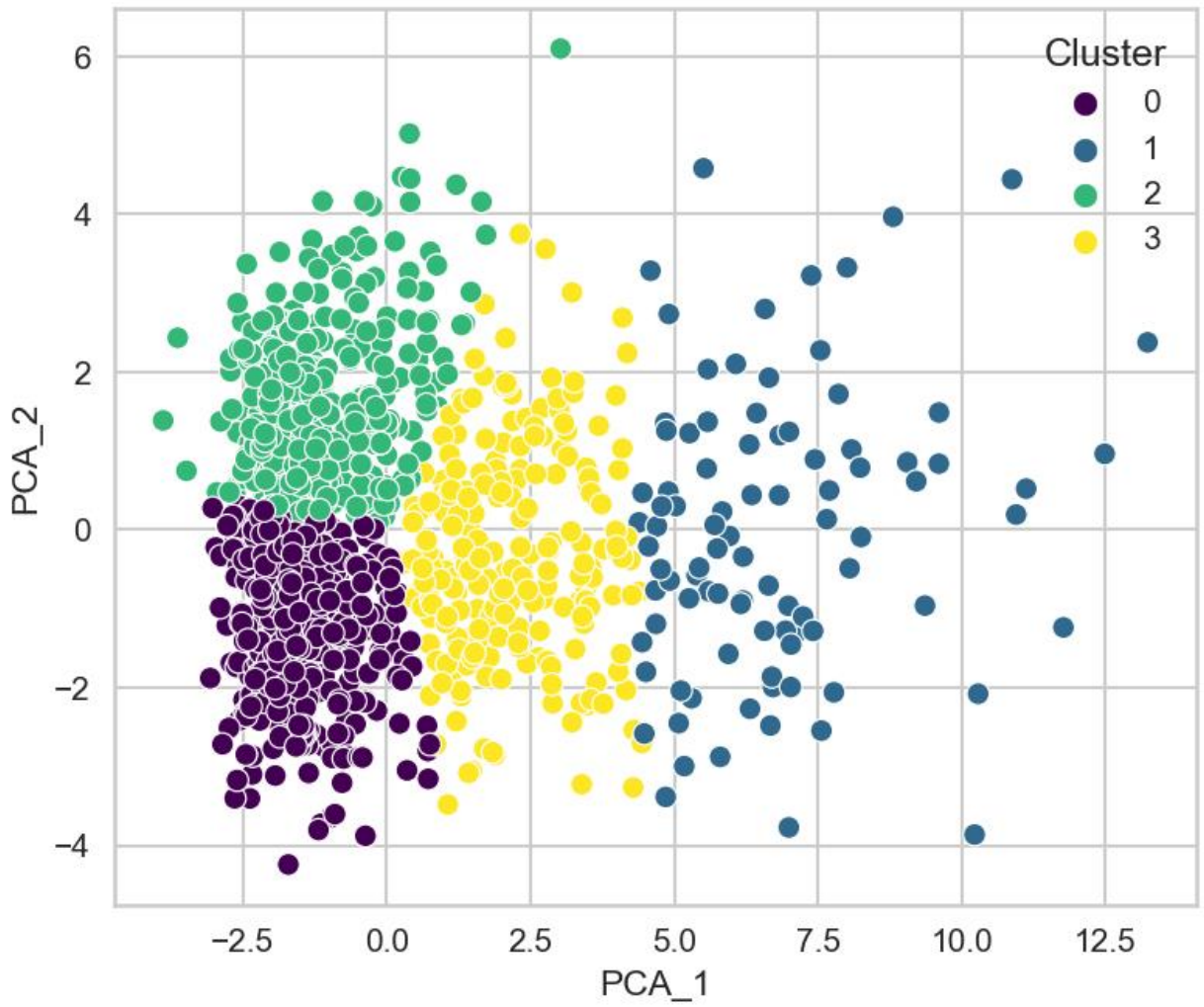
```
df_reduction_t
```

	PCA_1	PCA_2	Cluster
0	-1.556245	-1.176028	0
1	-1.581689	0.555600	2
2	4.905551	2.735371	1
3	-0.345759	-0.088841	0
4	-1.402452	-0.515081	0
...	...	...	...
1205	6.646240	1.928438	1
1206	-2.155633	2.647937	2
1207	-2.134485	0.242618	0
1208	8.806652	3.967336	1
1209	0.747920	-2.719591	0

1210 rows × 3 columns

Hình 68: Kết quả phân cụm bằng MiniBatch Kmeans

Trực quan hóa kết quả phân cụm bằng MiniBatchKMeans như dưới đây.



Hình 69: Kết quả trực quan hóa bằng MiniBatchKmeans

2.5 Thảo luận, phân tích, đánh giá và kết luận về kết quả nhận được sau khi tích dữ liệu

Từ ba mô hình đã được biểu thị ở trên, ta tính điểm silhouette_score để đánh giá độ hiệu quả của mô hình.

```python
from sklearn.metrics import silhouette_score

silhouette_avg_1 = silhouette_score(df_reduction1.iloc[:, :-1], df_reduction1['Cluster'])
silhouette_avg_2 = silhouette_score(df_reduction2.iloc[:, :-1], df_reduction2['cluster'])
silhouette_avg_3 = silhouette_score(df_reduction_t.iloc[:, :-1], df_reduction_t['Cluster'])
print(f"Điểm Silhouette mô hình Kmeans: {silhouette_avg_1:.2f}")
print(f"Điểm Silhouette mô hình DBScan: {silhouette_avg_2:.2f}")
print(f"Điểm Silhouette mô hình MiniBatchKmeans: {silhouette_avg_3:.2f}")
```

Python

```
Điểm Silhouette mô hình Kmeans: 0.37
Điểm Silhouette mô hình DBScan: 0.07
Điểm Silhouette mô hình MiniBatchKmeans: 0.39
```

Hình 70: Mã nguồn và kết quả kiểm tra của các mô hình

Dựa trên điểm Silhouette, mô hình K-Means (0.37) và MiniBatchKMeans (0.39) đều cho thấy chất lượng phân cụm tốt hơn so với mô hình DBScan (0.07). Điểm Silhouette là một đánh giá về mức độ đồng nhất và tách biệt giữa các cụm, và điểm cao hơn thường chỉ ra rằng phân cụm của mô hình đó là tốt.

Vì vậy, nếu mục tiêu là phân loại dữ liệu thành các nhóm một cách chất lượng và hiệu quả, có thể ưa thích sử dụng mô hình K-Means hoặc MiniBatchKMeans thay vì DBScan trong trường hợp này.

K-Means là một phương pháp phân cụm đơn giản nhưng mạnh mẽ, thích hợp cho dữ liệu có cấu trúc hình cụm tròn và có kích thước tương đồng. Nó tốt cho việc phân chia dữ liệu thành các nhóm với đồng nhất cao.

MiniBatchKMeans là một biến thể của K-Means thích hợp cho dữ liệu lớn và có thể cung cấp sự đồng nhất tương đối tốt (0.39) trong trường hợp của bạn.

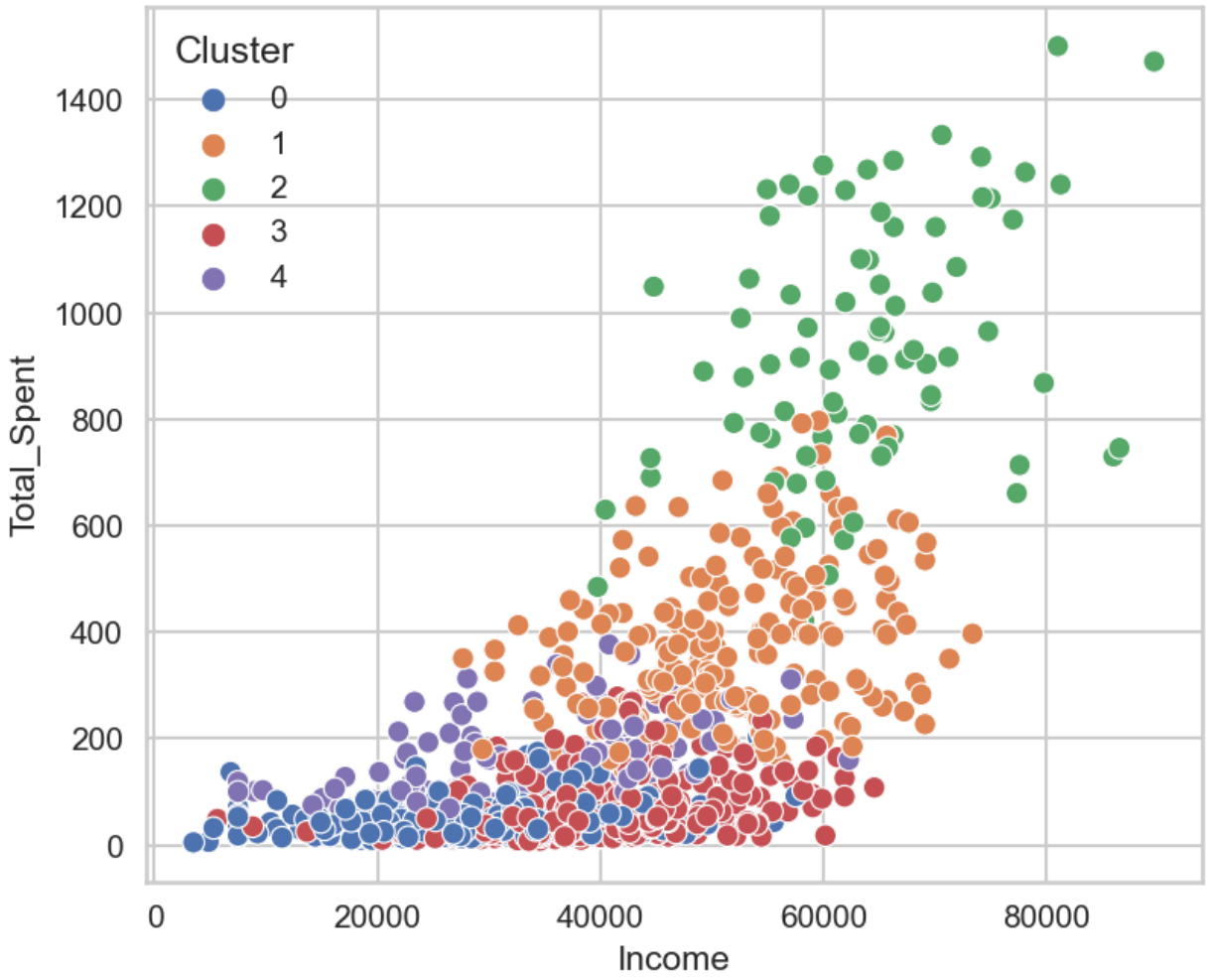
Trong khi đó, DBScan (0.07) có thể không phù hợp với dữ liệu hoặc yêu cầu các điều kiện về mật độ và hình dạng cụm khác nhau để hoạt động tốt. DBSCan đã không đạt được mức độ đồng nhất cao như K-Means.

Do đó, dựa trên đánh giá Silhouette, khuyến nghị sử dụng K-Means hoặc MiniBatchKMeans để phân tích dữ liệu và tạo các cụm trong dữ liệu này.

Qua biểu đồ nhiệt được vẽ ở trên, chúng tôi chọn ra 2 cột có độ tương quan khá cao là 'Income' ,'Total_Spent' , sử dụng nó để chiếu và phân cụm về 5 cụm có Tổng thu nhập và tổng chi tiêu khác nhau.

```
plt.figure(figsize=(6,5),dpi=150)
sns.scatterplot(x=df['Income'],y=df['Total_Spent'],hue=df['Cluster'], palette='deep')
```

Hình 71: Mã nguồn trực quan phân cụm dựa trên tổng thu và chi



Hình 72: Trực quan phân cụm dựa vào 2 tổng thu và tổng chi

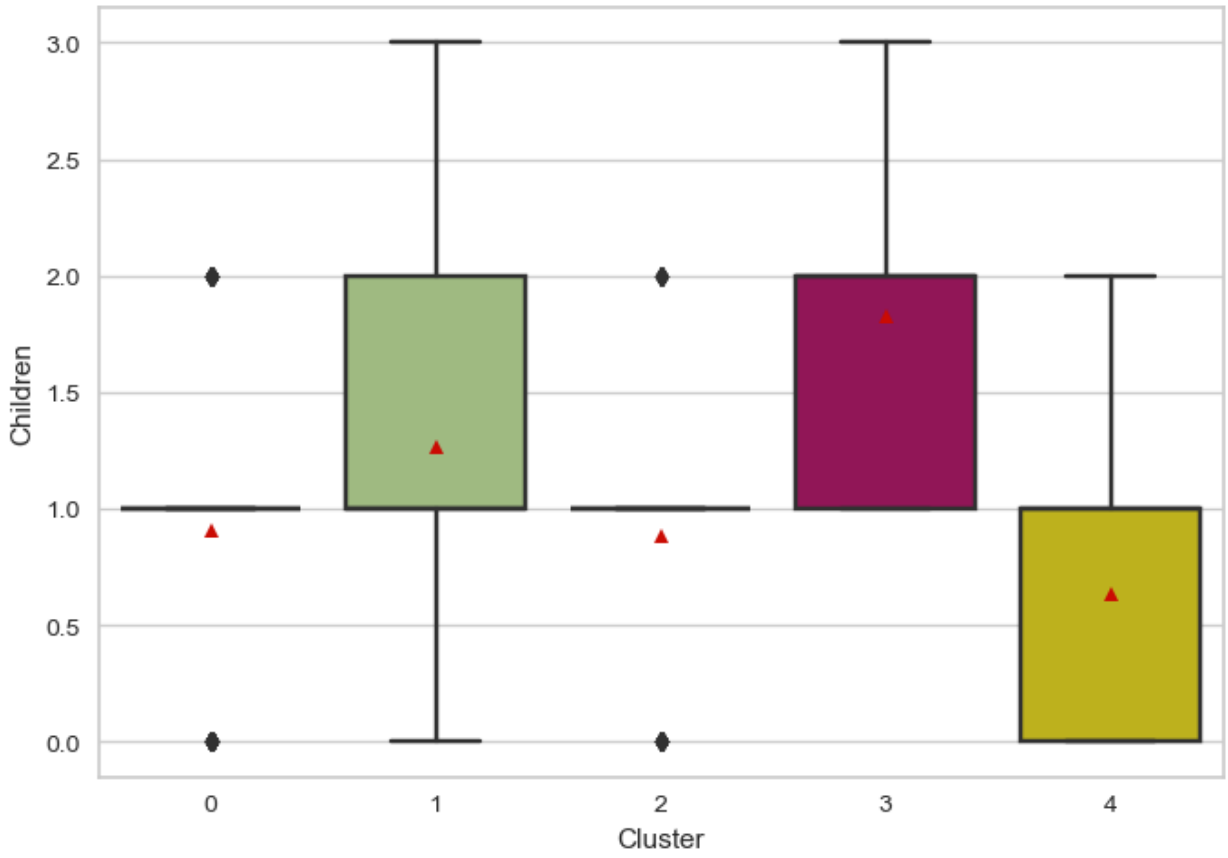
Từ biểu đồ trên, chúng tôi phân tích cụm dựa trên thu nhập và tổng chi tiêu như sau:

* Cluster 0 : Thu nhập thấp & Chi tiêu thấp.
* Cluster 1 : Thu nhập trung bình & Chi tiêu trung bình đến cao.
* Cluster 2 : Thu nhập Cao & Chi tiêu cao.

* Cluster 3 : Thu nhập trung bình đến cao & Chi tiêu thấp.
* Cluster 4 : Thu nhập thấp & Chi tiêu thấp đến trung bình.

Tương tự với biểu đồ boxplot để cho thấy số lượng con cái của từng cụm:

```
sns.boxplot(y=df['Children'],x=df['Cluster'],showmeans=True)
```



Hình 73: Biểu đồ hộp về số lượng trẻ em của mỗi cụm

Giải thích:

* Cluster 0 : Đa số có 1 con cái
* Cluster 1 : Đa số có 1-2 con cái
* Cluster 2 : Đa số có 1 con cái
* Cluster 3 : Đa số có 1-2 con cái
* Cluster 4 : Đa số có 0-1 con cái

TÀI LIỆU THAM KHẢO