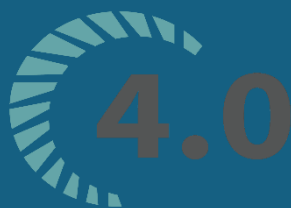


TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ
MINH

BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

BÁO CÁO ĐỒ ÁN CUỐI KỲ



PHÂN TÍCH DỮ LIỆU ỨNG DỤNG

Mục lục

A.	Bảng thông tin chi tiết nhóm	2
B.	Bảng phân chia công việc	3
C.	Yêu cầu của bài tập	4
D.	Kết quả	4
	Kiểm tra dữ liệu missing, duplicates, outliers	5
	Kiểm tra cân bằng giữa nhấn click/không click quảng cáo	9
	Xử lý dữ liệu missing, duplicates, outliers	11
	Phân tích đơn biến (Univariate Analysis), Bivariate Analysis	16
	Kết luận các điểm chính quan sát được từ dữ liệu sau khi phân tích Univariate và Bivariate.....	19
	Quan sát nhận xét về Thời gian sử dụng website theo độ tuổi, theo thu nhập hoặc theo thành phố sinh sống; Thời điểm trong ngày (sáng, trưa, chiều, tối); Chủ đề quảng cáo được quan tâm nhất hoặc ít được quan tâm nhất.....	31
	Dự đoán khả năng 1 người dùng có chọn xem quảng cáo được hiển thị hay không (lựa chọn và cài đặt thuật toán, cài đặt ít nhất 2 thuật toán để so sánh với nhau).....	39
	Đánh giá chất lượng mô hình sử dụng cross validation với các độ đo precision, recall, f1 trên tập train, test. Kết luận mô hình được lựa chọn.	52

A. Bảng thông tin chi tiết nhóm

Mã nhóm:	01
Tên nhóm:	01
Số lượng:	03

MSSV	Họ tên	Email	Điện thoại	Hình ảnh
20127201	Trương Bảo Khang	20127201@student.hcmus.edu.vn		
20127579	Lâm Kim Nhân	20127579@student.hcmus.edu.vn		
20127189	Nguyễn Quốc Huy	20127189@student.hcmus.edu.vn		

B. Bảng phân chia công việc

Công việc thực hiện	Người thực hiện	Mức độ hoàn thành	Đánh giá của nhóm
<ul style="list-style-type: none"> Phân tích đơn biến (Univariate Analysis), Bivariate Analysis Kết luận các điểm chính quan sát được từ dữ liệu sau khi phân tích đơn biến (Univariate Analysis), Bivariate Analysis Quan sát và nhận xét về: <ul style="list-style-type: none"> Thời gian sử dụng website theo độ tuổi, theo thu nhập hoặc theo thành phố sinh sống. Thời điểm trong ngày (sáng, trưa, chiều, tối) Chủ đề quảng cáo được quan tâm nhất hoặc ít được quan tâm nhất Dự đoán khả năng 1 người dùng có chọn xem quảng cáo được hiển thị hay không (lựa chọn và cài đặt thuật toán, cài đặt ít nhất 2 thuật toán để so sánh với nhau) Đánh giá chất lượng mô hình sử dụng cross validation với 	Trương Bảo Khang	100%	10/10

<ul style="list-style-type: none"> các độ đo precision, recall, f1 trên tập train, test. Kết luận mô hình được lựa chọn Viết báo cáo 			
<ul style="list-style-type: none"> Kiểm tra dữ liệu missing, duplicates, outliers Xử lý dữ liệu missing, duplicates, outliers Kiểm tra sự cân bằng giữa nhãn: click/không click quảng cáo 	Lâm Kim Nhân	100%	10/10
<ul style="list-style-type: none"> Kiểm tra dữ liệu missing, duplicates, outliers Xử lý dữ liệu missing, duplicates, outliers Kiểm tra sự cân bằng giữa nhãn: click/không click quảng cáo 	Nguyễn Quốc Huy	100%	10/10

C. Yêu cầu của bài tập

Loại bài tập	<input type="checkbox"/> Lý thuyết <input checked="" type="checkbox"/> Thực hành <input checked="" type="checkbox"/> Đồ án <input type="checkbox"/> Bài tập
Ngày bắt đầu	
Ngày kết thúc	

D. Kết quả

Kiểm tra dữ liệu missing, duplicates, outliers

Đầu tiên ta sẽ khai phá dữ liệu trước, tìm hiểu số lượng cột và dòng trong tập dữ liệu, tìm hiểu kiểu dữ liệu của từng thuộc tính.

```
[ ] print(df.shape)
    print(df.columns)
```

```
⇒ (1002, 10)
   Index(['Daily Time Spent on Site', 'Age', 'Area Income',
         'Daily Internet Usage', 'Ad Topic Line', 'City', 'Male', 'Country',
         'Timestamp', 'Clicked on Ad'],
        dtype='object')
```

```
[ ] df.info()
```

```
⇒ <class 'pandas.core.frame.DataFrame'>
   RangeIndex: 1002 entries, 0 to 1001
   Data columns (total 10 columns):
    #   Column                                  Non-Null Count  Dtype
   ---  -
    0   Daily Time Spent on Site                1002 non-null   float64
    1   Age                                      1001 non-null   float64
    2   Area Income                             1002 non-null   float64
    3   Daily Internet Usage                    1002 non-null   float64
    4   Ad Topic Line                           1002 non-null   object
    5   City                                    1002 non-null   object
    6   Male                                    1002 non-null   int64
    7   Country                                 1002 non-null   object
    8   Timestamp                               1002 non-null   object
    9   Clicked on Ad                           1002 non-null   int64
   dtypes: float64(4), int64(2), object(4)
   memory usage: 78.4+ KB
```

Sau đó ta sẽ đếm số lượng giá trị riêng biệt của từng thuộc tính, để có thể hiểu rõ hơn và biết được cách phù hợp để phân tích tập dữ liệu, cũng như để tìm được có biến phụ thuộc nào trong tập dữ liệu không. Ở đây ta có biến “Clicked on Ad” là biến nhị phân, nên ta có thể sử dụng biến nhị phân làm biến phụ thuộc và tất cả các biến còn lại sẽ là biến độc lập.

```
[ ] for column in df.columns:  
    num_distinct_values = len(df[column].unique())  
    print(f"{column} -> {num_distinct_values} distinct values\n")
```

```
⇒ Daily Time Spent on Site -> 900 distinct values
```

```
Age -> 44 distinct values
```

```
Area Income -> 1000 distinct values
```

```
Daily Internet Usage -> 966 distinct values
```

```
Ad Topic Line -> 1000 distinct values
```

```
City -> 969 distinct values
```


```
Male -> 2 distinct values
```


```
Country -> 237 distinct values
```

```
Timestamp -> 997 distinct values
```

```
Clicked on Ad -> 2 distinct values
```

Sau đó ta sẽ kiểm tra có thuộc tính nào trong tập dữ liệu xuất hiện giá trị Null không. Ở đây thì ta có thể thấy cột Age có 1 giá trị Null.


 `df.isnull().sum()`

 `0`

Daily Time Spent on Site	0
Age	1
Area Income	0
Daily Internet Usage	0
Ad Topic Line	0
City	0
Male	0
Country	0
Timestamp	0
Clicked on Ad	0

dtype: int64

Nếu như có thì ta sẽ tiếp tục tìm chi tiết vị trí của giá trị Null. Khi cột Age có giá trị Null thì ta sẽ tìm chi tiết dòng chứa giá trị Null của cột Age.

 `df[df.isnull().any(axis=1)]`

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
1000	45.01	NaN	29875.8	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	3/6/2016 21:43	1

Tiếp tục kiểm tra có dữ liệu trùng lặp không. Nếu có thì in ra dòng bị trùng lặp.

`df[df.duplicated(keep=False)]`

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
727	81.17	30.0	57195.96	231.91	Optimized static archive	Dayton	1	United States Minor Outlying Islands	3/6/2016 7:00	0
755	81.17	30.0	57195.96	231.91	Optimized static archive	Dayton	1	United States Minor Outlying Islands	3/6/2016 7:00	0
817	36.91	48.0	54645.20	159.69	Ameliorated coherent open architecture	North Samantha	0	Zimbabwe	24/2/2016 7:13	1
1001	36.91	48.0	54645.20	159.69	Ameliorated coherent open architecture	North Samantha	0	Zimbabwe	24/2/2016 7:13	1

Tiếp tục kiểm tra có giá trị ngoại lai trong từng thuộc tính không. Ở đây ta sẽ lấy các thuộc tính có kiểu dữ liệu số để kiểm tra vì dữ liệu kiểu số là dạng dữ liệu xảy ra outliers nhiều, và ta sử dụng biểu đồ boxplot để kiểm tra.

```
numerical_columns = ['Daily Time Spent on Site', 'Age', 'Area Income', 'Daily Internet Usage']

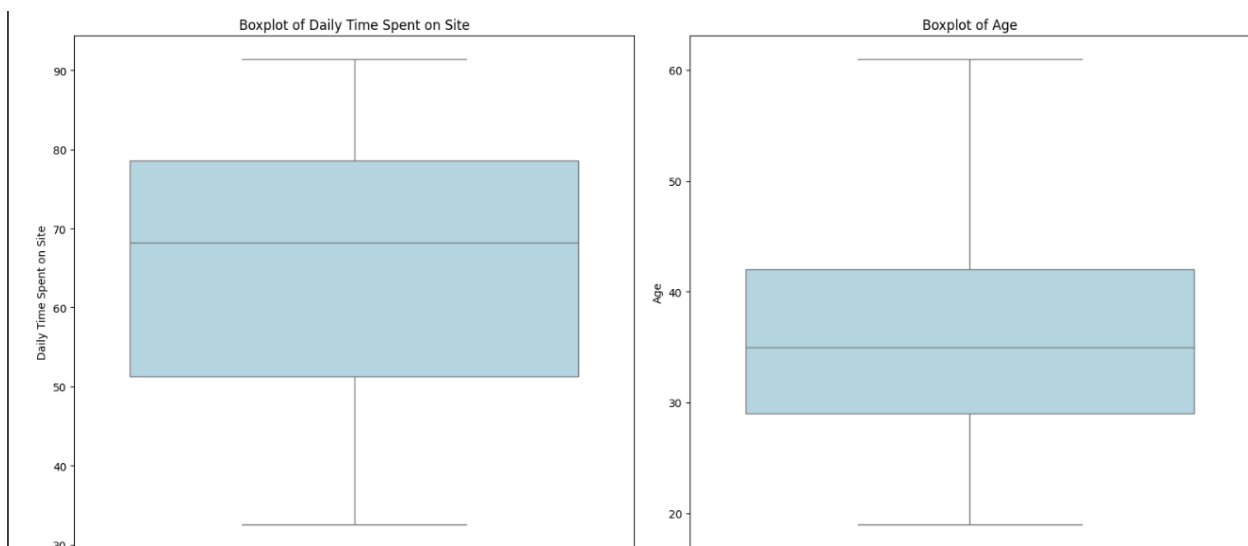
cols = 2
rows = math.ceil(len(numerical_columns) / cols)

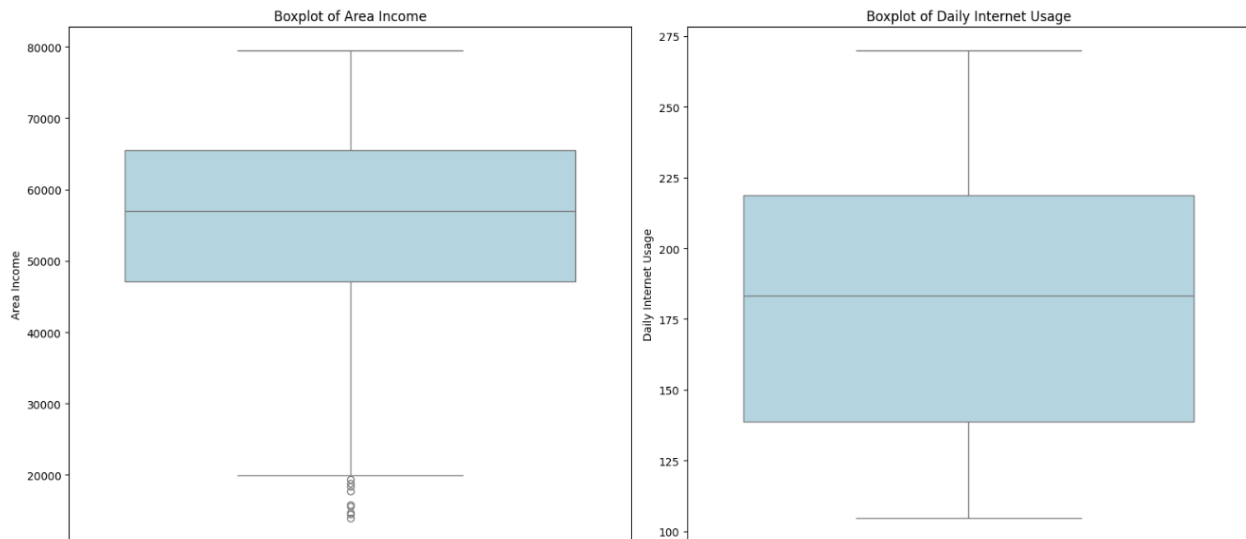
plt.figure(figsize=(16, 14))
for i, column in enumerate(numerical_columns):
    plt.subplot(rows, cols, i + 1)

    sns.boxplot(y=df[column], color='lightblue')

    plt.title(f'Boxplot of {column}', fontsize=12)

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





Kiểm tra cân bằng giữa nhấn click/không click quảng cáo

Để kiểm tra sự cân bằng giữa nhấn click (tức 1) và nhấn không click (tức 0) trong cột Clicked on Ad, ta chỉ cần sử dụng hàm `value_counts()` và đếm số lượng giá trị 0 và 1 xuất hiện trong cột Clicked on Ad, nếu số lượng giá trị xuất hiện của 2 giá trị 0 và 1 đều bằng nhau thì có thể kết luận nhấn click và không click đã cân bằng. Ngoài ra để chắc chắn thì nhóm sẽ tính thêm phần trăm tỷ lệ xuất hiện của 2 nhấn click và không click, sau đó sẽ tính độ chênh lệch tuyệt đối của cả 2 nhấn, và đặt ngưỡng là 5% (tức chỉ chênh lệch khoảng 50 dòng). Nếu dưới 5% thì kết luận nhấn click và không click gần như cân bằng, bởi vì sẽ có trường hợp là số lượng xuất hiện của 2 nhấn lần lượt là 498 và 502 hoặc 490 và 510, nếu vượt quá ngưỡng 5% thì sẽ dẫn đến không cân bằng.

```

value_counts = df['Clicked on Ad'].value_counts()
print(value_counts)

total = len(df['Clicked on Ad'])
proportion_0 = value_counts[0] / total
proportion_1 = value_counts[1] / total

print(proportion_0)
print(proportion_1)

if abs(proportion_0 - proportion_1) < 0.05:
    print("Nhấn click và không click cân bằng.")
else:
    print("Nhấn click và không click không cân bằng.")

```

```

Clicked on Ad
0      501
1      501
Name: count, dtype: int64
0.5
0.5
Nhấn click và không click cân bằng.

```

Xử lý dữ liệu missing, duplicates, outliers

Xử lý dữ liệu missing, lúc đầu ta đã kiểm tra và có giá trị missing trong cột Age. Để xử lý missing, ta sẽ kiểm tra nên sử dụng dữ liệu Mean, Median hay Mode để điền khuyết vào giá trị missing. Khi kiểm tra 3 giá trị thì ta thấy giá trị Median phù hợp nhất trong 3 giá trị nên ta sẽ lấy giá trị Median thêm vào chỗ bị thiếu

```
print(df['Age'].mean())
print(df['Age'].median())
print(df['Age'].mode())

df['Age'].fillna(df['Age'].median(), inplace=True)
```

36.02497502497503
35.0
0 31.0
Name: Age, dtype: float64
<ipython-input-14-b39c2cd16c11>:5: FutureWarning: A value is
The behavior will change in pandas 3.0. This inplace method w
For example, when doing 'df[col].method(value, inplace=True)'
df['Age'].fillna(df['Age'].median(), inplace=True)

Sau đó ta sẽ kiểm tra lại 1 lần nữa để chắc chắn giá trị missing không còn.

```
df.isnull().sum()

0
Daily Time Spent on Site 0
Age 0
Area Income 0
Daily Internet Usage 0
Ad Topic Line 0
City 0
Male 0
Country 0
Timestamp 0
Clicked on Ad 0

dtype: int64
```

Tiếp tục xử lý giá trị duplicated, khi này ta đã kiểm tra có 2 dòng bị trùng lặp dữ liệu, ta sẽ xóa đi 2 dòng dữ liệu đó và kiểm tra lại để chắc chắn không còn giá trị duplicated. Lúc đầu khi ta khai phá tập dữ liệu có 1002 dòng, sau khi xóa dữ liệu duplicated bây giờ tập dữ liệu còn 1000 dòng.

```
df_unique = df.drop_duplicates()
df_unique[df_unique.duplicated(keep=False)]
df_unique
```

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35.0	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	27/3/2016 0:53	0
1	80.23	31.0	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	4/4/2016 1:39	0
2	69.47	26.0	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	13/3/2016 20:35	0
3	74.15	29.0	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	10/1/2016 2:31	0
4	68.37	35.0	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	3/6/2016 3:36	0
...
996	72.97	30.0	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	11/2/2016 21:49	1
997	51.30	45.0	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	22/4/2016 2:07	1
998	51.63	51.0	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	1/2/2016 17:24	1
999	55.55	19.0	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	24/3/2016 2:35	0
1000	45.01	35.0	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	3/6/2016 21:43	1

1000 rows x 10 columns

Vì việc xóa dữ liệu duplicated đồng nghĩa với việc sự cân bằng của nhãn click và không click cũng sẽ bị ảnh hưởng nên ta cần phải kiểm tra lại 1 lần nữa để đảm bảo cân bằng vẫn còn.

```

value_counts = df_unique['Clicked on Ad'].value_counts()
print(value_counts)

total = len(df_unique['Clicked on Ad'])
proportion_0 = value_counts[0] / total
proportion_1 = value_counts[1] / total

print(proportion_0)
print(proportion_1)

if abs(proportion_0 - proportion_1) < 0.05:
    print("Nhãn click và không click cân bằng.")
else:
    print("Nhãn click và không click không cân bằng.")

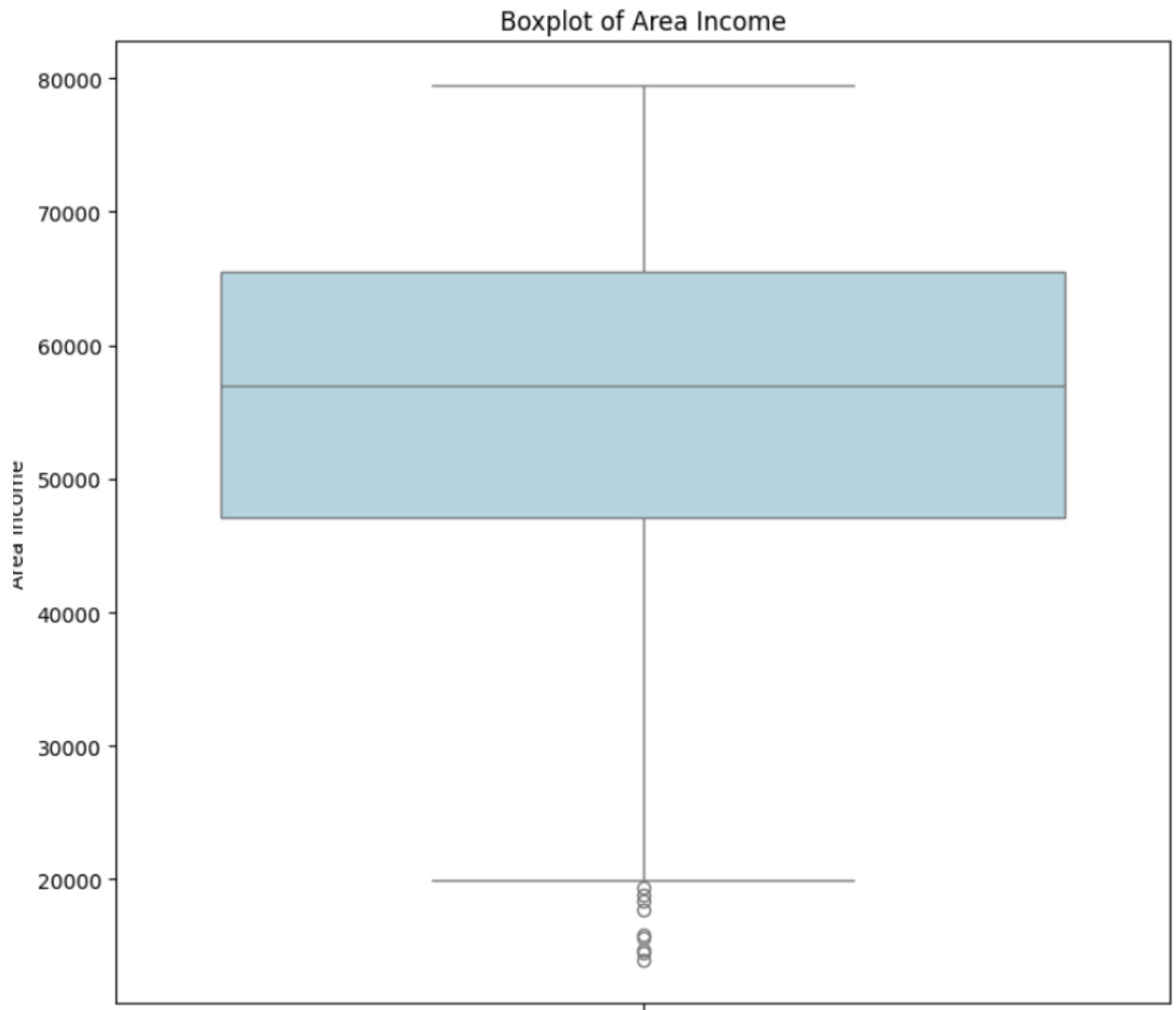
```

```

Clicked on Ad
0      500
1      500
Name: count, dtype: int64
0.5
0.5
Nhãn click và không click cân bằng.

```

Về việc xử lý dữ liệu outliers, chỉ có dữ liệu về Area Income xuất hiện nhiều outliers ở khoảng dưới 20000, nhưng do đây là dữ liệu về Thu nhập nên có thể xét đến khía cạnh thực tế, vẫn có người dân có thu nhập dưới 20000 nên sẽ không bỏ dữ liệu outliers này.



Phân tích đơn biến (Univariate Analysis), Bivariate Analysis

Phân tích đơn biến các thuộc tính dạng số, ta sẽ dùng biểu đồ histogram và boxplot

```
numeric_cols = ['Daily Time Spent on Site', 'Age', 'Area Income', 'Daily Internet Usage']
for col in numeric_cols:
    # Histogram
    plt.figure(figsize=(10, 4))
    sns.histplot(df_unique[col], bins=20, kde=True)
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()

    # Box Plot
    plt.figure(figsize=(10, 2))
    sns.boxplot(x=df_unique[col])
    plt.title(f'Box Plot of {col}')
    plt.xlabel(col)
    plt.show()
```

Đối với các thuộc tính phân loại thì ta dùng biểu đồ pie chart

```
categorical_cols = ['City', 'Country']

for col in categorical_cols:

    top_values = df_unique[col].value_counts().nlargest(10)

    plt.figure(figsize=(8, 8))
    plt.pie(
        top_values,
        labels=top_values.index,
        autopct='%1.1f%%',
        startangle=140
    )
    plt.title(f'Distribution of Top 10 {col}')
    plt.show()
```

Phân tích Bivariate :

Để phân tích tần suất tỷ lệ nhấp chuột theo độ tuổi, ta sử dụng biểu đồ cột thể hiện tần suất kết hợp với biểu đồ boxplot


```
plt.figure(figsize=(12, 6))
sns.countplot(x='Age', hue='Clicked on Ad', data=df_unique, palette='viridis')
plt.title('Tần suất nhấp chuột theo độ tuổi')
plt.xlabel('Độ tuổi')
plt.ylabel('Số lượng')
plt.xticks(rotation=45)
plt.show()
```

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Clicked on Ad', y='Age', data=df_unique, palette='viridis')
plt.title('Sự phân phối độ tuổi theo hành vi nhấp vào quảng cáo')
plt.xlabel('Clicked on Ad (0: Không, 1: Có)')
plt.ylabel('Age')
plt.grid(True)
plt.show()
```

Để phân tích được tần suất tỷ lệ nhấp chuột theo mức thu nhập, ta cần phải nhóm các giá trị thu nhập theo từng nhóm thu nhập (10k-20k, 20k-30k...), đồng thời tạo thêm cột mới để lưu và sau đó thực hiện vẽ biểu đồ để phân tích

```
bins = [10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000]
labels = ['10k-20k', '20k-30k', '30k-40k', '40k-50k', '50k-60k', '60k-70k', '70k-80k']

# Tạo một cột mới với các khoảng
df_unique['Income Group'] = pd.cut(df['Area Income'], bins=bins, labels=labels, right=False)
df_unique
```

Ở đây ta cũng sẽ sử dụng biểu đồ cột thể hiện tần suất kết hợp với biểu đồ boxplot

```
plt.figure(figsize=(12, 6))
sns.countplot(x='Income Group', hue='Clicked on Ad', data=df_unique, palette='viridis')
plt.title('Tần suất nhấp vào quảng cáo theo thu nhập')
plt.xlabel('Thu nhập')
plt.ylabel('Tần suất nhấp vào quảng cáo')
plt.xticks(rotation=45)
plt.show()
```

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Clicked on Ad', y='Area Income', data=df_unique, palette='viridis')
plt.title('Sự phân phối thu nhập theo hành vi nhấp vào quảng cáo')
plt.xlabel('Clicked on Ad (0: Không, 1: Có)')
plt.ylabel('Area Income')
plt.grid(True)
plt.show()
```

Đối với phân tích tần suất tỷ lệ nhấp chuột theo giới tính, ta cũng sử dụng biểu đồ cột

```
plt.figure(figsize=(12, 6))
sns.countplot(x='Male', hue='Clicked on Ad', data=df_unique, palette='viridis')
plt.title('Tần suất nhấp chuột theo giới tính')
plt.xlabel('Giới tính')
plt.ylabel('Tần suất nhấp chuột')
plt.xticks(rotation=45)
plt.show()
```

Ngoài ra, nhóm có làm thêm về phân tích tần suất tỷ lệ nhấp chuột qua từng tháng. Để phân tích được thì ta sẽ tách dữ liệu trong cột Timestamp ra thành 2 cột gồm ngày và giờ để dễ trực quan hơn, đồng thời phục vụ cho các yêu cầu sau.

```
df_unique[['Date', 'Time']] = df_unique['Timestamp'].str.split(' ', expand=True)
df_unique
```

Sau đó ta sẽ lấy cột Date và đổi định dạng sang kiểu dữ liệu ngày giờ và lấy dữ liệu tháng trong cột Date và cho vào cột mới, sau đó sẽ tiến hành trực quan.

```
df_unique['Date'] = pd.to_datetime(df_unique['Date'], errors='coerce')
df_unique['Date'] = df_unique['Date'].dt.date
df_unique['Month'] = pd.to_datetime(df_unique['Date']).dt.month

plt.figure(figsize=(12, 6))
sns.countplot(x='Month', hue='Clicked on Ad', data=df_unique, palette='viridis')
plt.title('Tần suất nhấp chuột qua từng tháng')
plt.xlabel('Thu nhập')
plt.ylabel('Tần suất nhấp chuột')
plt.xticks(rotation=45)
plt.show()
```

Để kiểm tra sự tương quan giữa các thuộc tính đối với thuộc tính nhấp quảng cáo, ta dùng ma trận tương quan và trực quan bằng biểu đồ heatmap. Đối với ma trận tương quan thì chỉ có thể sử dụng dữ liệu dạng số để trực quan.

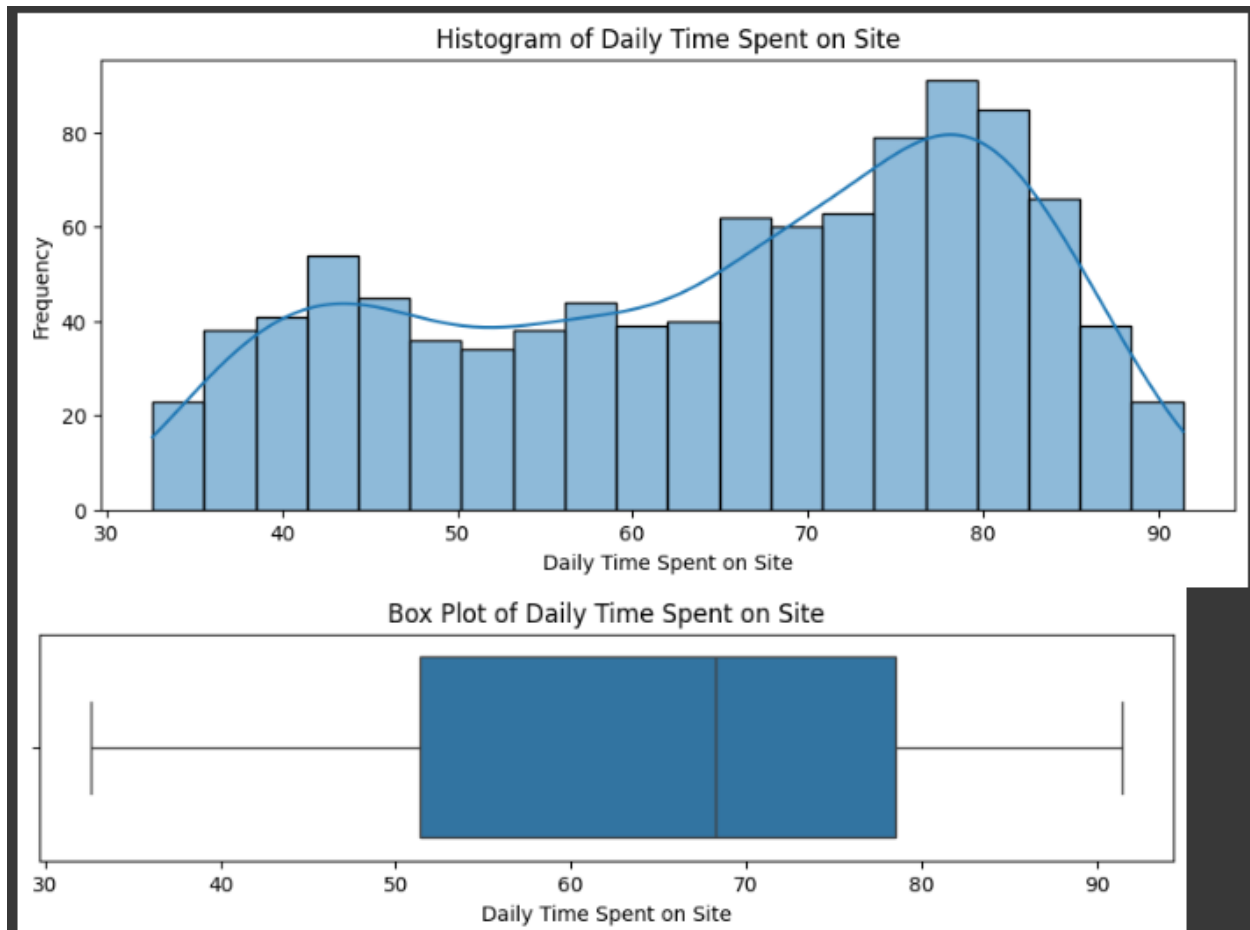
```
numerical_df = df_unique.select_dtypes(include=['float64', 'int64'])

correlation_matrix = numerical_df.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', square=True)
plt.title('Ma trận tương quan giữa các thuộc tính')
plt.show()
```

Kết luận các điểm chính quan sát được từ dữ liệu sau khi phân tích Univariate và Bivariate

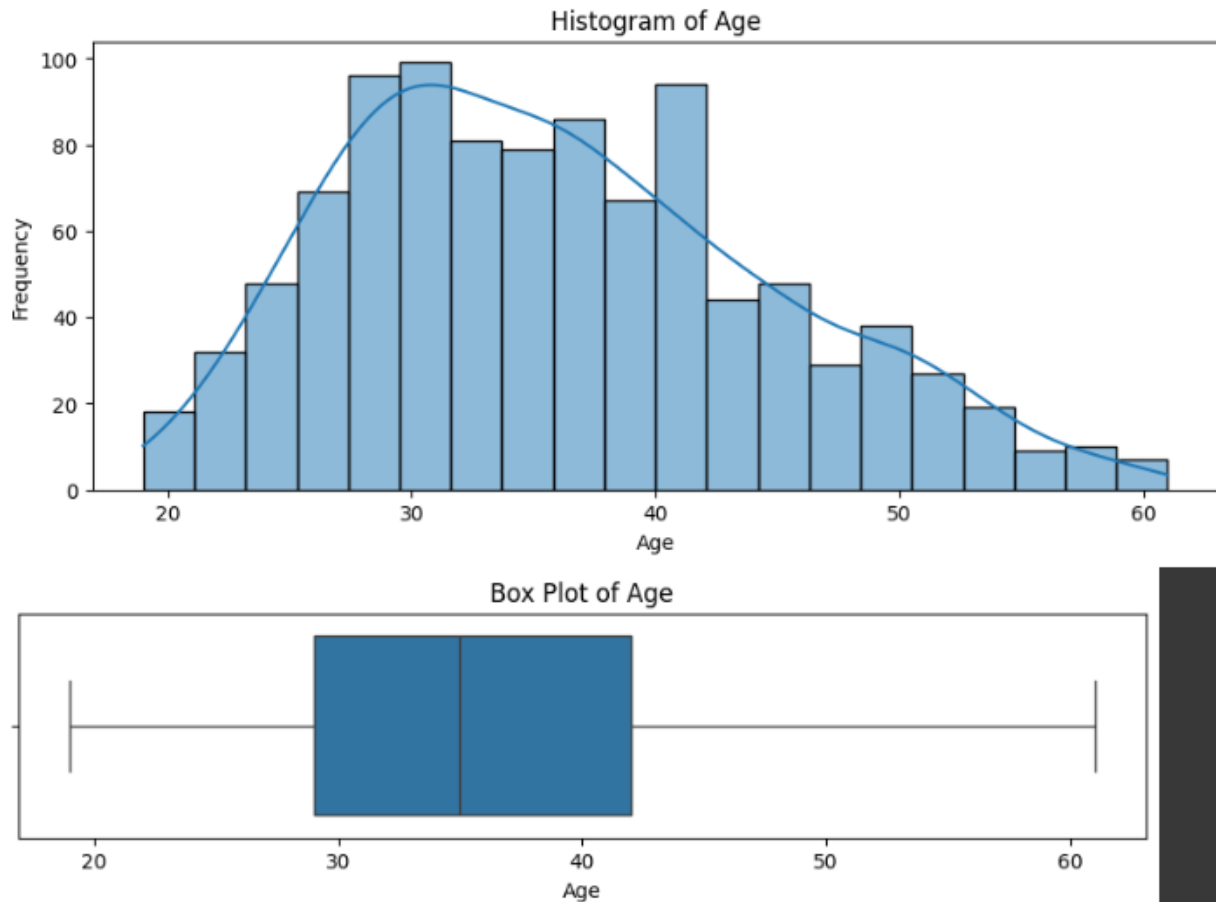
Phân tích đơn biến :



Dựa vào biểu đồ có thể thấy, phân phối dữ liệu có dạng gần với phân phối chuẩn lệch phải nhẹ, với phần đuôi kéo dài về phía các giá trị lớn hơn.

Khoảng thời gian 70-80 phút có tần suất cao nhất, cho thấy đa số người dùng của tập dữ liệu dành thời gian này trên trang web.

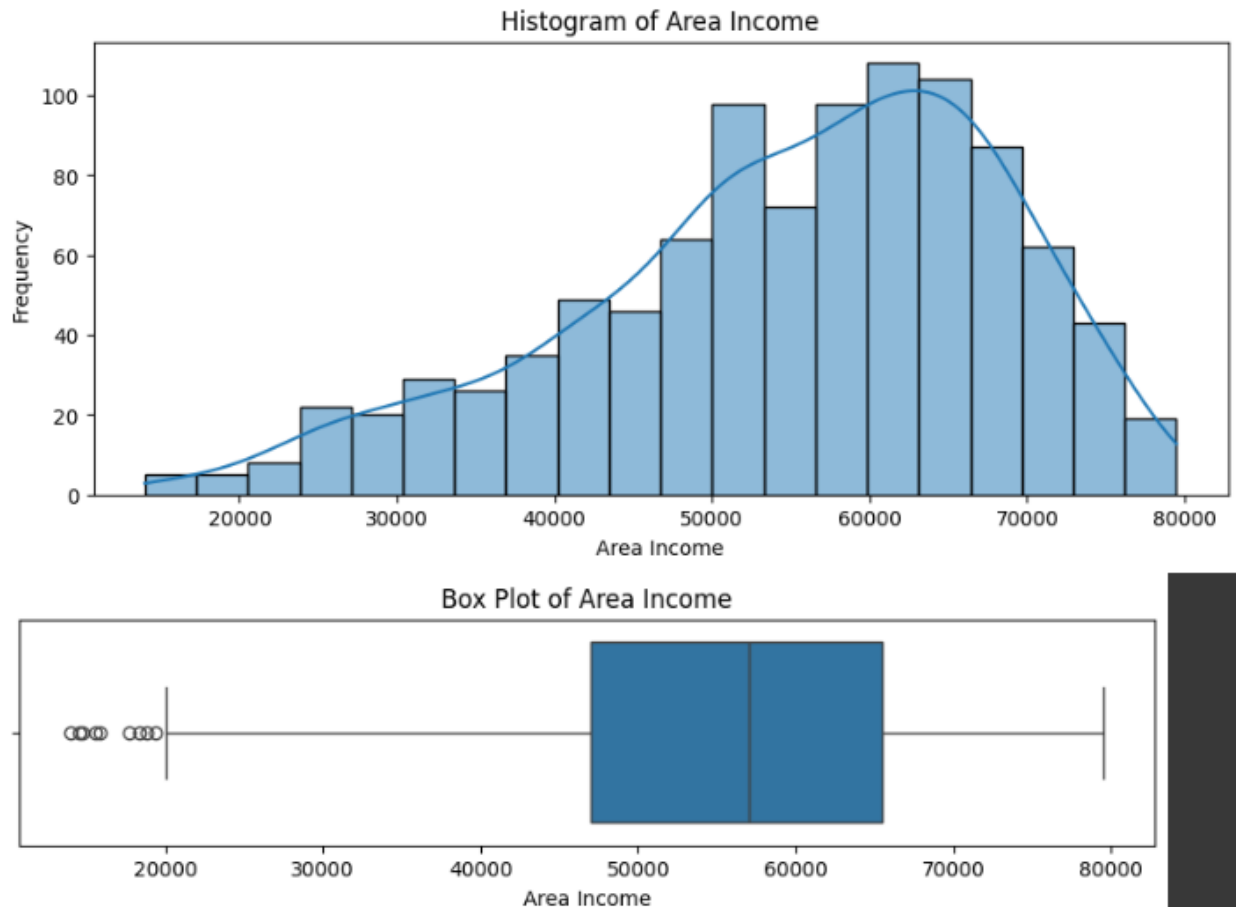
Không có outlier đáng kể, hộp trung vị cho thấy phần lớn dữ liệu tập trung trong khoảng từ 50 đến 80 phút.



Dựa vào biểu đồ, phân phối dữ liệu có dạng phân phối chuẩn lệch trái, với phần đầu kéo về phía giá trị nhỏ hơn.

Độ tuổi gần 30 đến 40 tuổi có tần suất cao nhất, cho thấy có xu hướng dành thời gian sử dụng internet nhiều hơn.

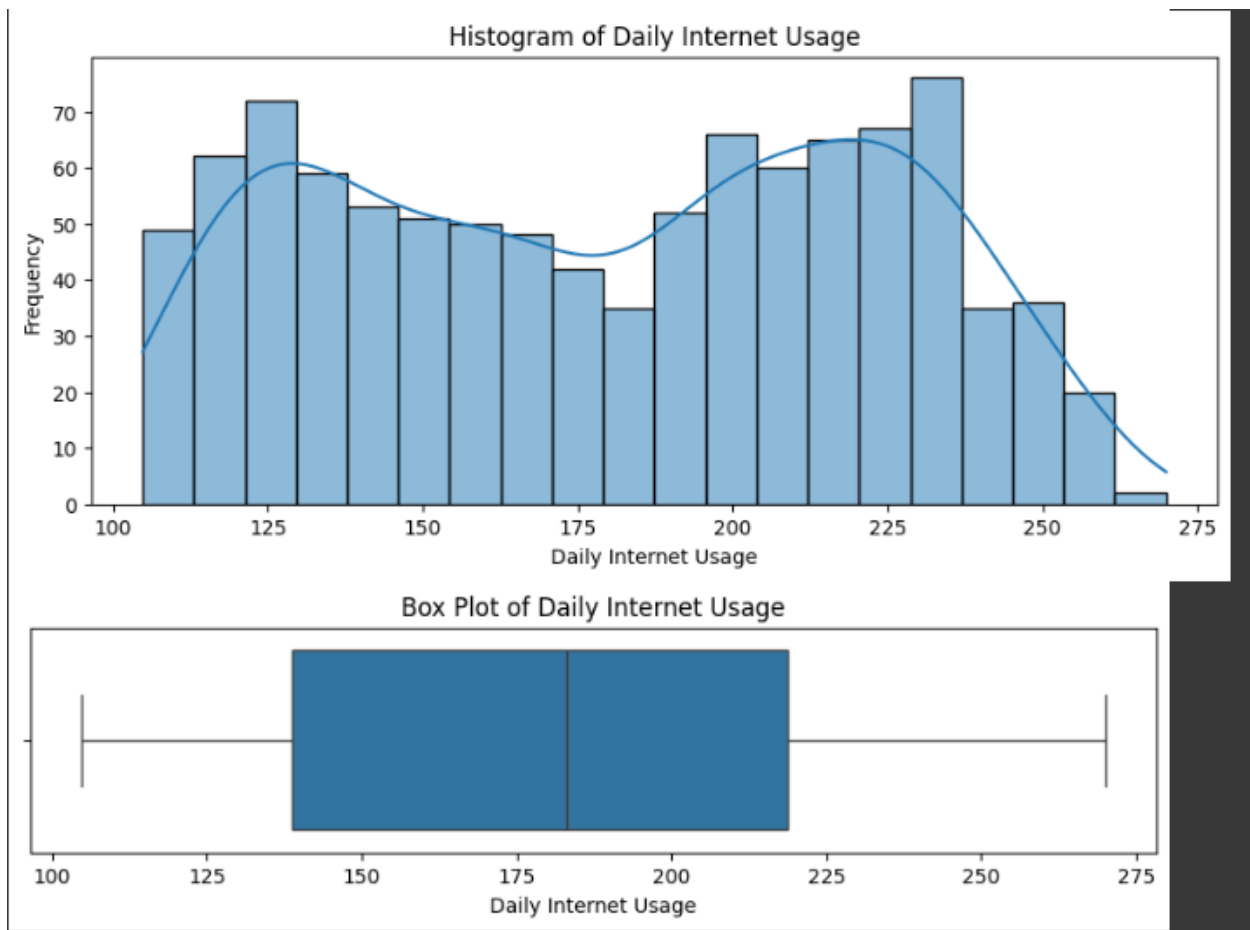
Không có outlier, hộp trung vị cho thấy phần lớn dữ liệu tập trung khoảng từ 30 đến 40 tuổi, cho thấy đa số giới thanh niên từ 30 đến 40 tuổi của tập dữ liệu sử dụng internet nhiều.



Dựa vào biểu đồ có thể thấy, phân phối dữ liệu có dạng gần với phân phối chuẩn lệch phải, với phần đuôi kéo dài về phía các giá trị lớn hơn.

Thu nhập từ 50,000 đến khoảng gần 70,000 có tần suất cao nhất, cho thấy những người có thu nhập càng cao sẽ có xu hướng sử dụng Internet nhiều hơn.

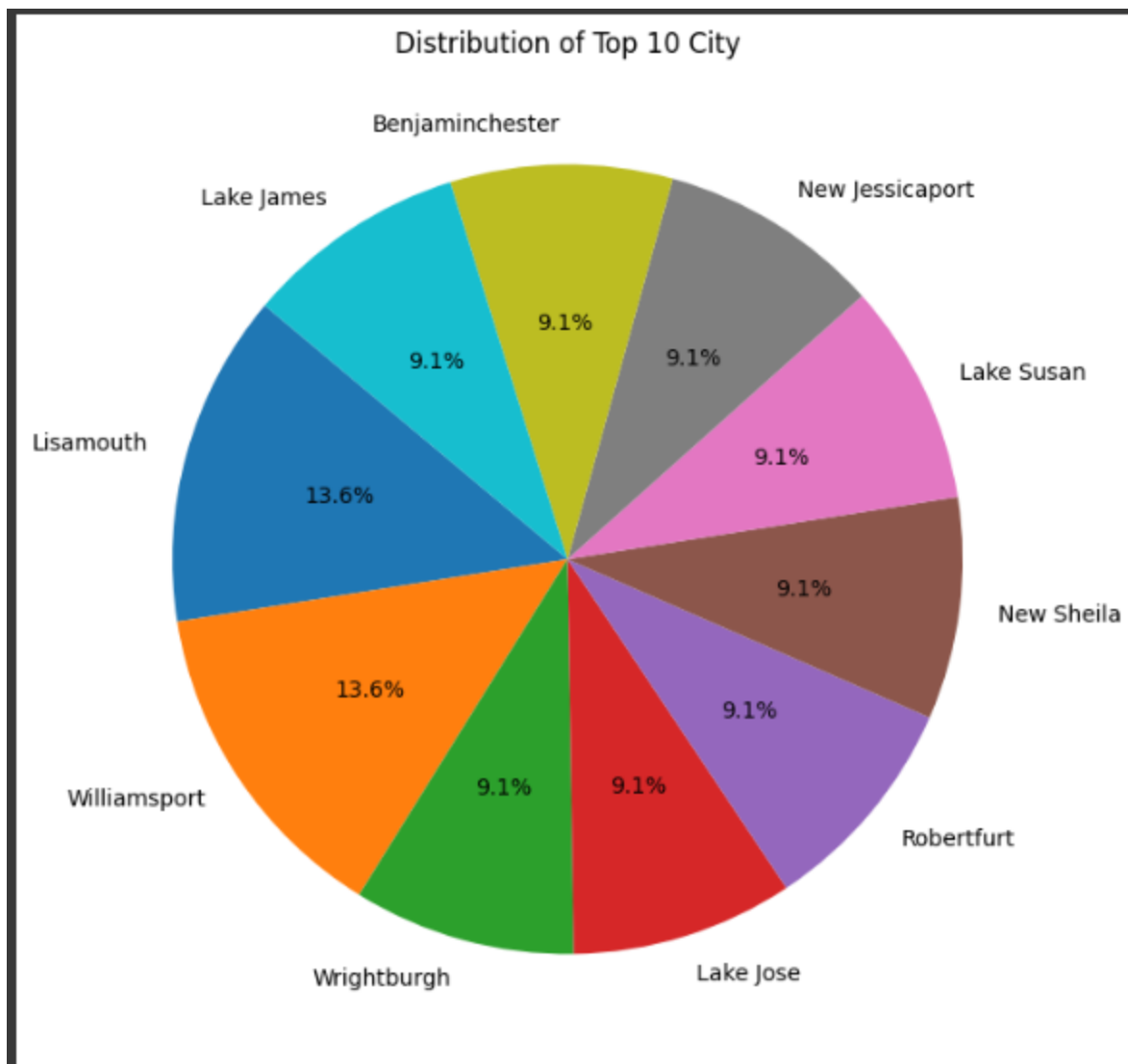
Có xuất hiện outlier, nhưng xét theo viễn cảnh thực tế vẫn có người có thu nhập dưới 20,000 nên vẫn chấp nhận được. Hộp trung vị cho thấy phần lớn dữ liệu tập trung ở khoảng 50,000-70,000, cho thấy đa số những người có mức thu nhập từ trung bình đến cao sẽ sử dụng internet nhiều.



Dựa vào biểu đồ có thể thấy, dữ liệu có xu hướng phân phối hai đỉnh (bimodal) với các tần suất cao tại khoảng 120-130 phút và 220-230 phút, có sự phân tán rộng, với phần đuôi dài ở phía giá trị lớn hơn.

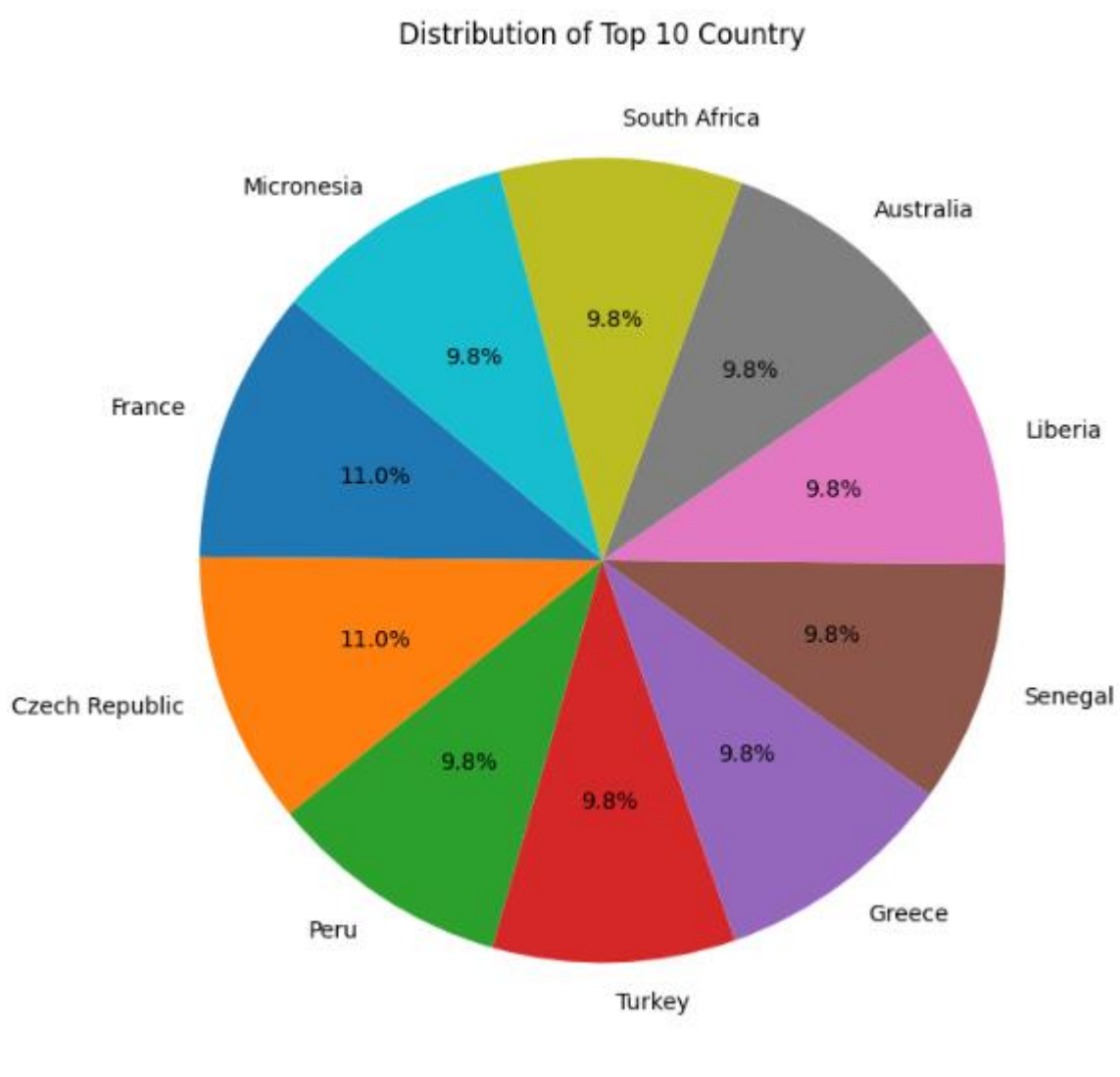
Khoảng giữa (IQR - Interquartile Range) tập trung từ 150 đến 230 phút, cho thấy phần lớn người dùng sẽ sử dụng internet từ 2 tiếng 30 phút đến gần 4 tiếng.

Dữ liệu cho thấy xu hướng hai cụm chính của thời gian sử dụng Internet hàng ngày: một nhóm sử dụng thấp (khoảng 120 phút) và một nhóm sử dụng cao hơn (khoảng 220 phút).



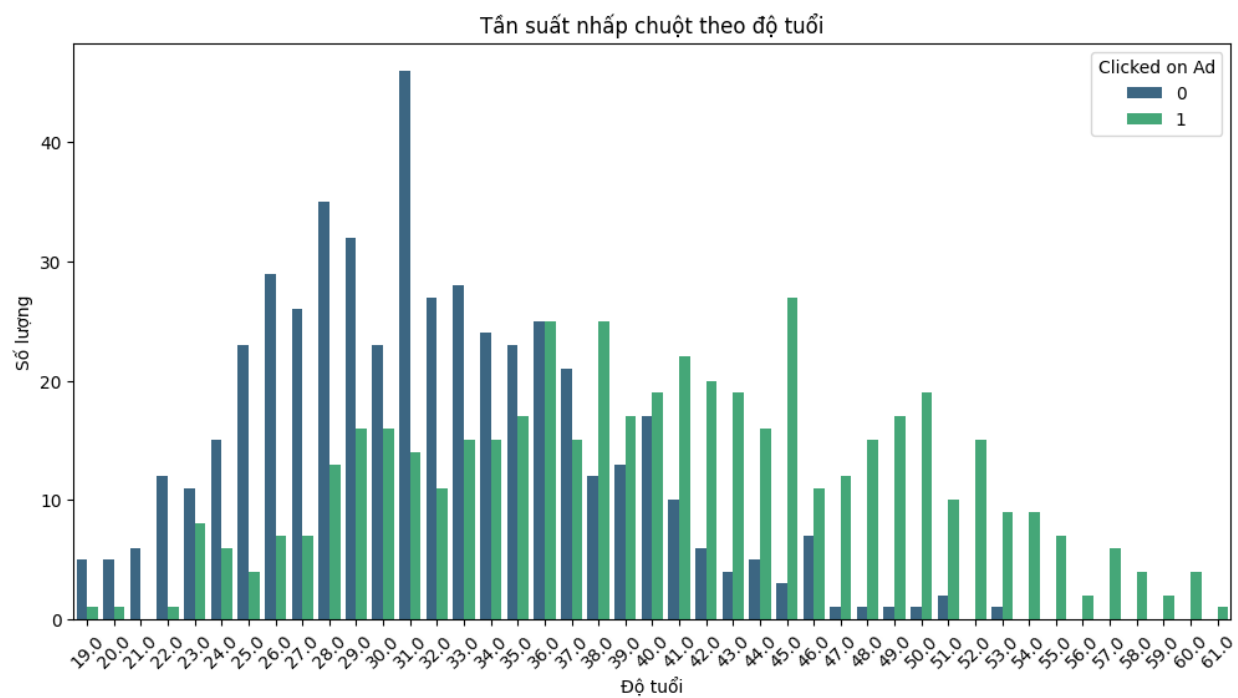
Do dữ liệu về thành phố trong tập dữ liệu có hơn 900 giá trị riêng biệt nên khó có thể trực quan cho toàn bộ 969 thành phố nên nhóm chỉ có thể trực quan và phân tích top 10 thành phố.

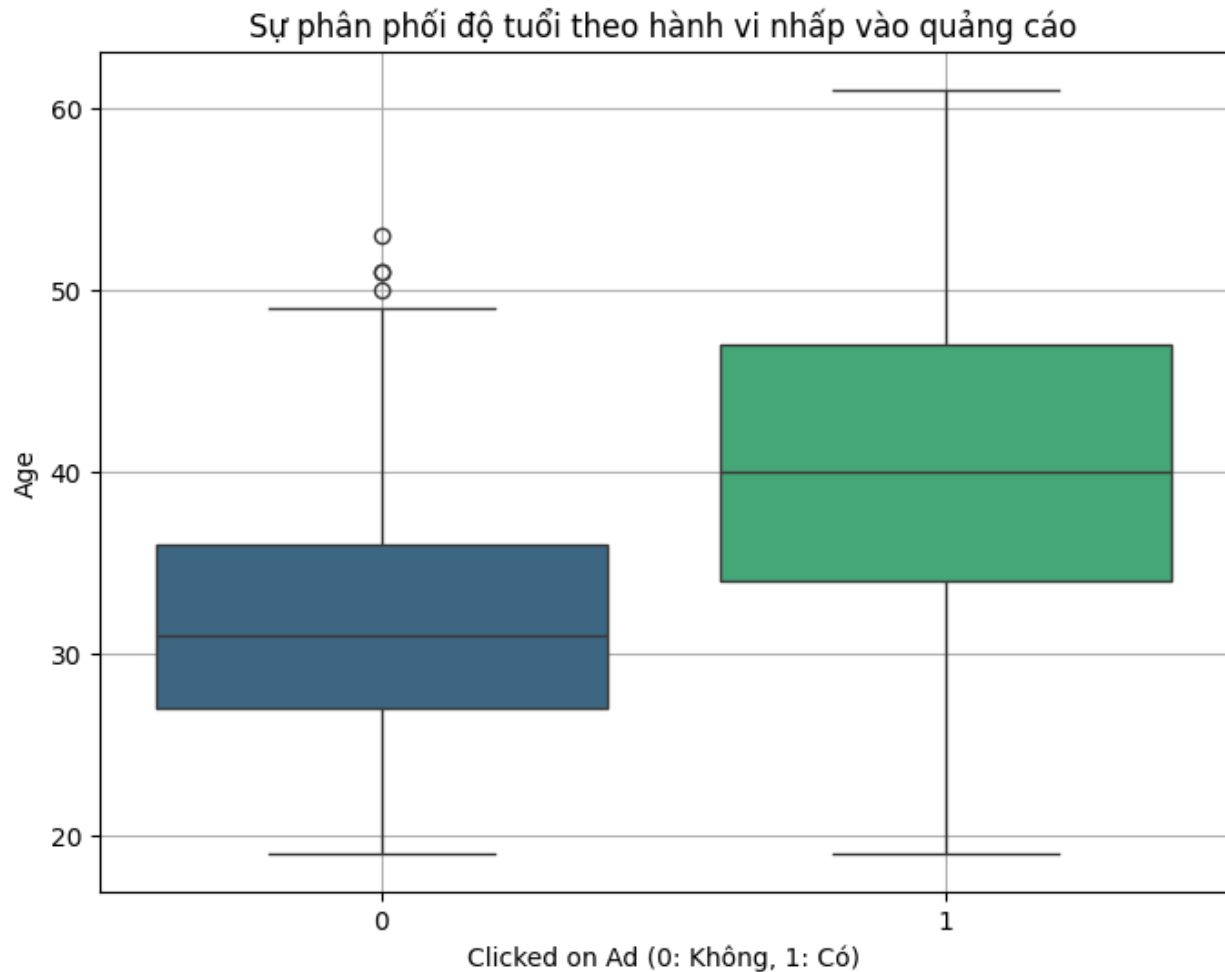
Dựa vào biểu đồ có thể thấy, thành phố Lisamouth và Williamsport chiếm tỷ lệ nhiều nhất với 13.6%, cho thấy đa số người sử dụng Internet đều cư trú tại 2 thành phố này, tiếp theo đó là các thành phố với tỉ lệ ngang nhau khoảng 9.1%



Đối với dữ liệu của các quốc gia cũng vậy, do có hơn 200 quốc gia riêng biệt trong tập dữ liệu nên nhóm cũng sẽ trực quan và phân tích top 10 quốc gia.

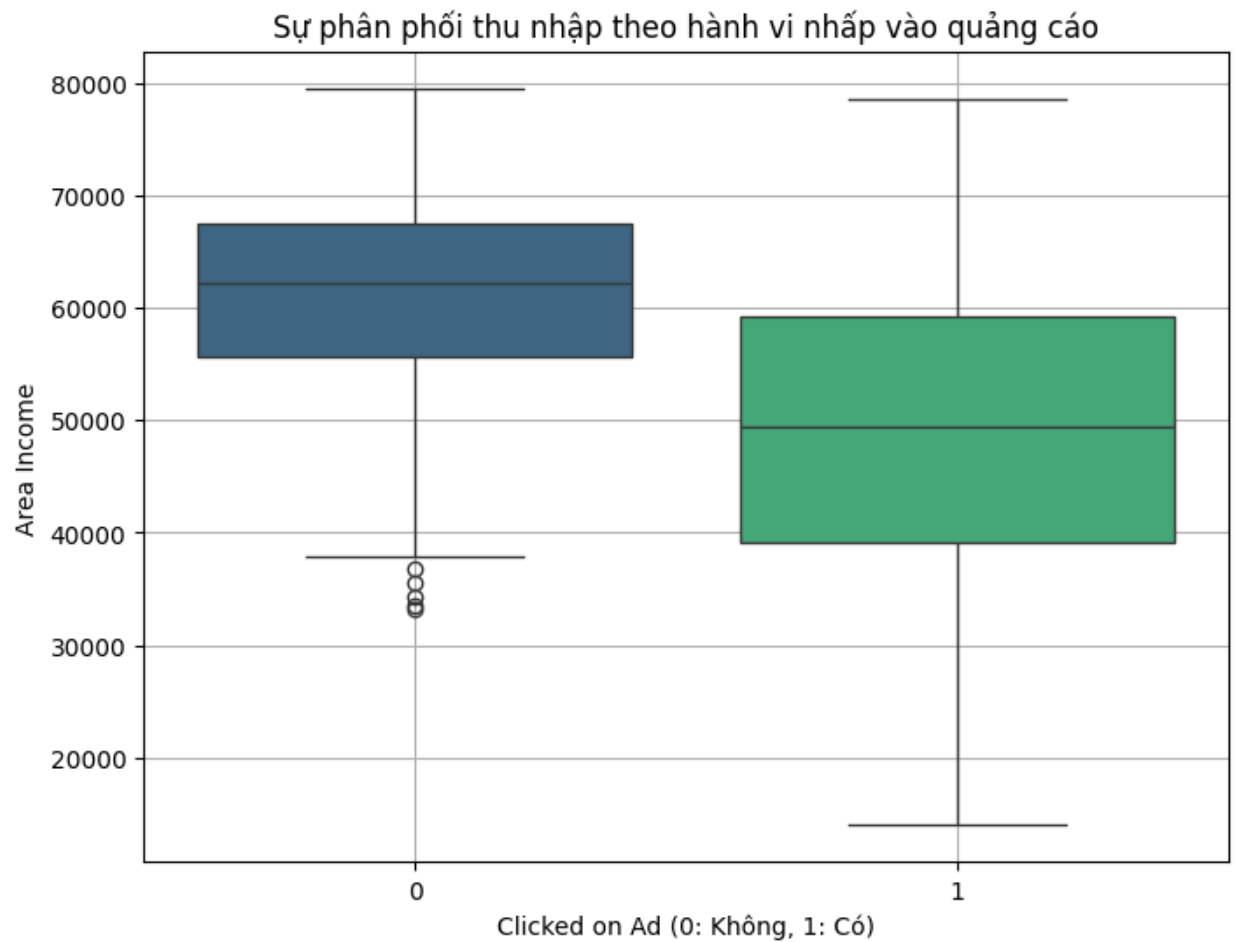
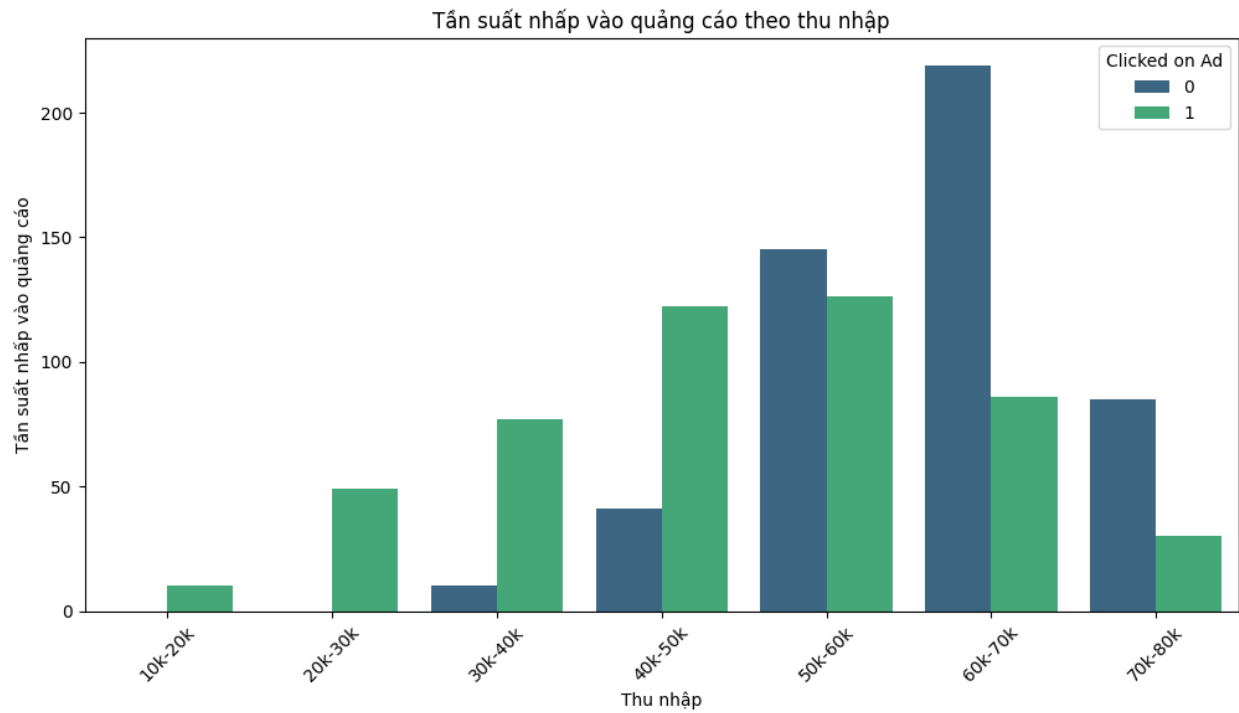
Dựa vào biểu đồ, 2 quốc gia là Pháp và Cộng hòa Séc chiếm tỷ lệ nhiều nhất khoảng 11%, cho thấy đa phần người sử dụng Internet có quốc tịch từ 2 quốc gia này, tiếp đó là các quốc gia như Úc, Nam Phi, Turkey... với tỷ lệ khoảng 9.8%





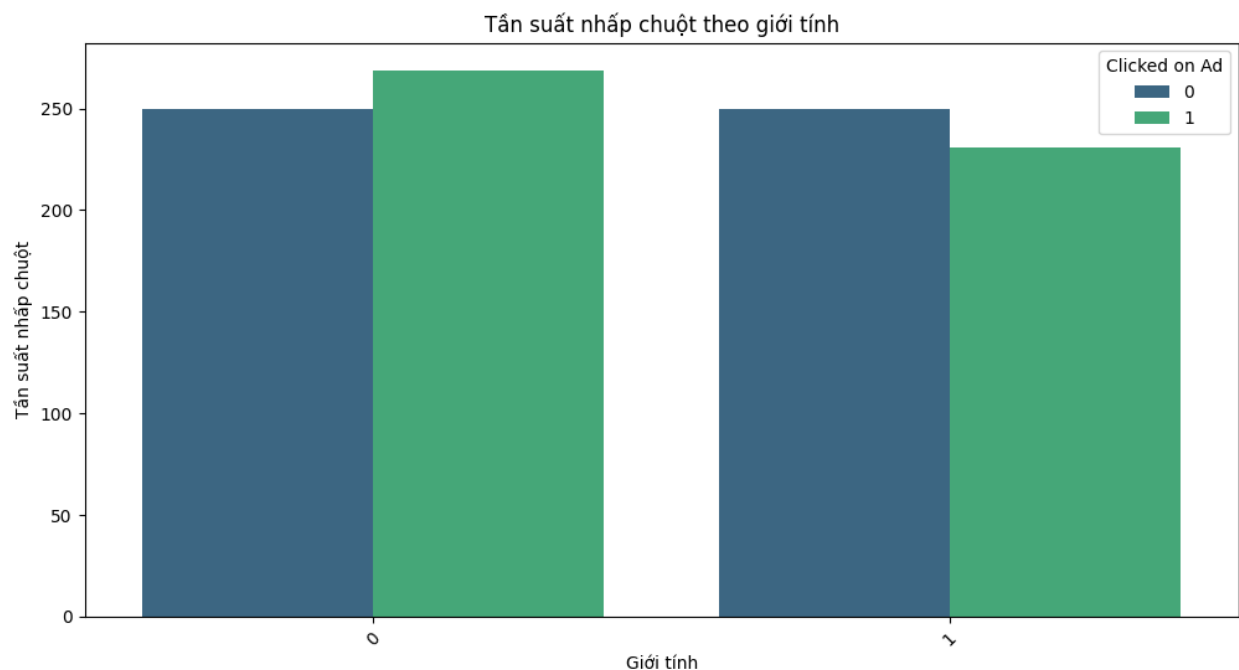
Đối với yêu cầu quan sát và phân tích tần suất nhấp chuột theo độ tuổi, theo 2 biểu đồ cột thể hiện tần suất và biểu đồ boxplot thể hiện phân phối độ tuổi, có thể thấy, ở độ tuổi 19 đến gần 40 tuổi đa số sẽ ít nhấp vào quảng cáo hơn, cũng có 1 vài trường hợp ngoại lai ở độ tuổi 47 đến 50, và cũng bắt đầu từ độ tuổi 23 cho đến 60 tuổi thì tỷ lệ nhấp vào quảng cáo xuất hiện nhiều hơn.

Có thể kết luận rằng xu hướng giới trẻ và những người độ tuổi thanh niên sẽ ít nhấp vào quảng cáo hơn, ngược lại những người ở độ tuổi trung niên sẽ nhấp vào quảng cáo nhiều hơn khi sử dụng Internet.

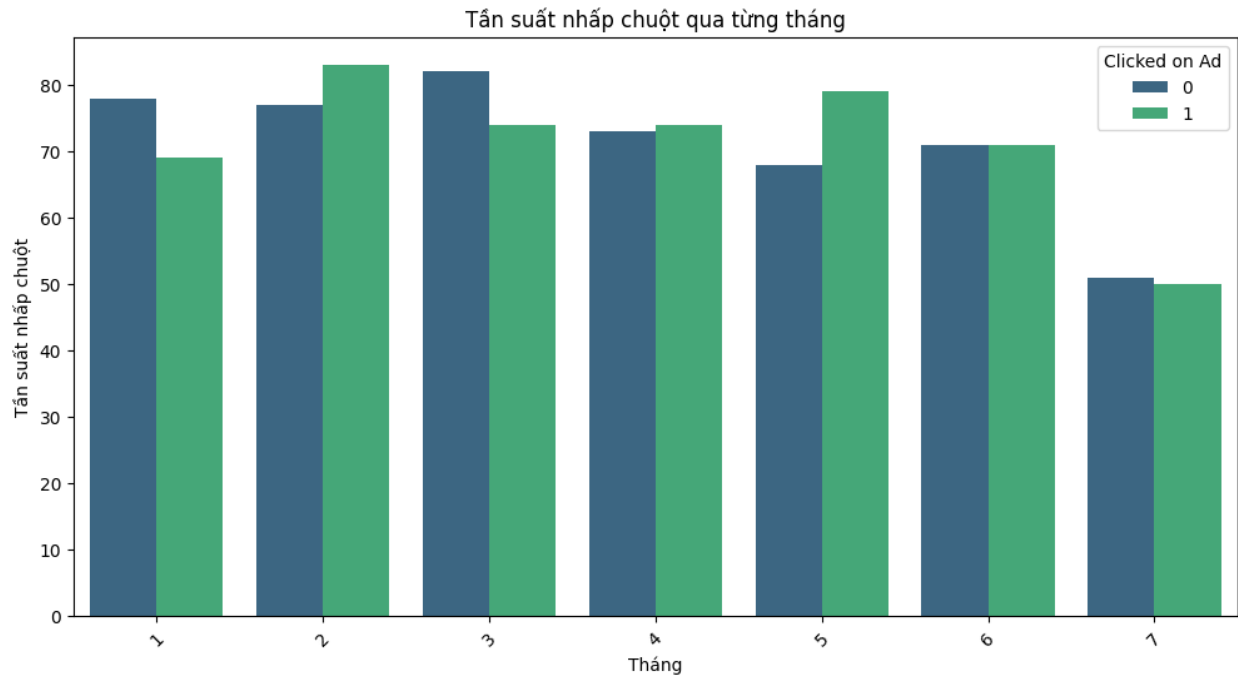


Đối với yêu cầu phân tích tần suất nhấp chuột theo mức thu nhập, lúc đầu trước khi thực hiện trực quan, nhóm đã có thực hiện biến đổi dữ liệu Area Income theo từng mức thu nhập khác nhau, để dễ trực quan hơn và dễ phân tích hơn. Theo 2 biểu đồ cột và boxplot, có thể thấy, đối với những người không nhấp vào quảng cáo phần lớn sẽ có mức thu nhập phân bố từ khoảng 50k đến 70k, trong khi đó những người nhấp vào quảng cáo sẽ có mức thu nhập phân bố từ khoảng 10k đến 60k.

Từ đó có thể kết luận, những người có mức thu nhập càng cao thì sẽ ít tương tác với quảng cáo hơn so với những người có mức thu nhập thấp. Những người có mức thu nhập thấp sẽ quan tâm và nhấp quảng cáo nhiều hơn.

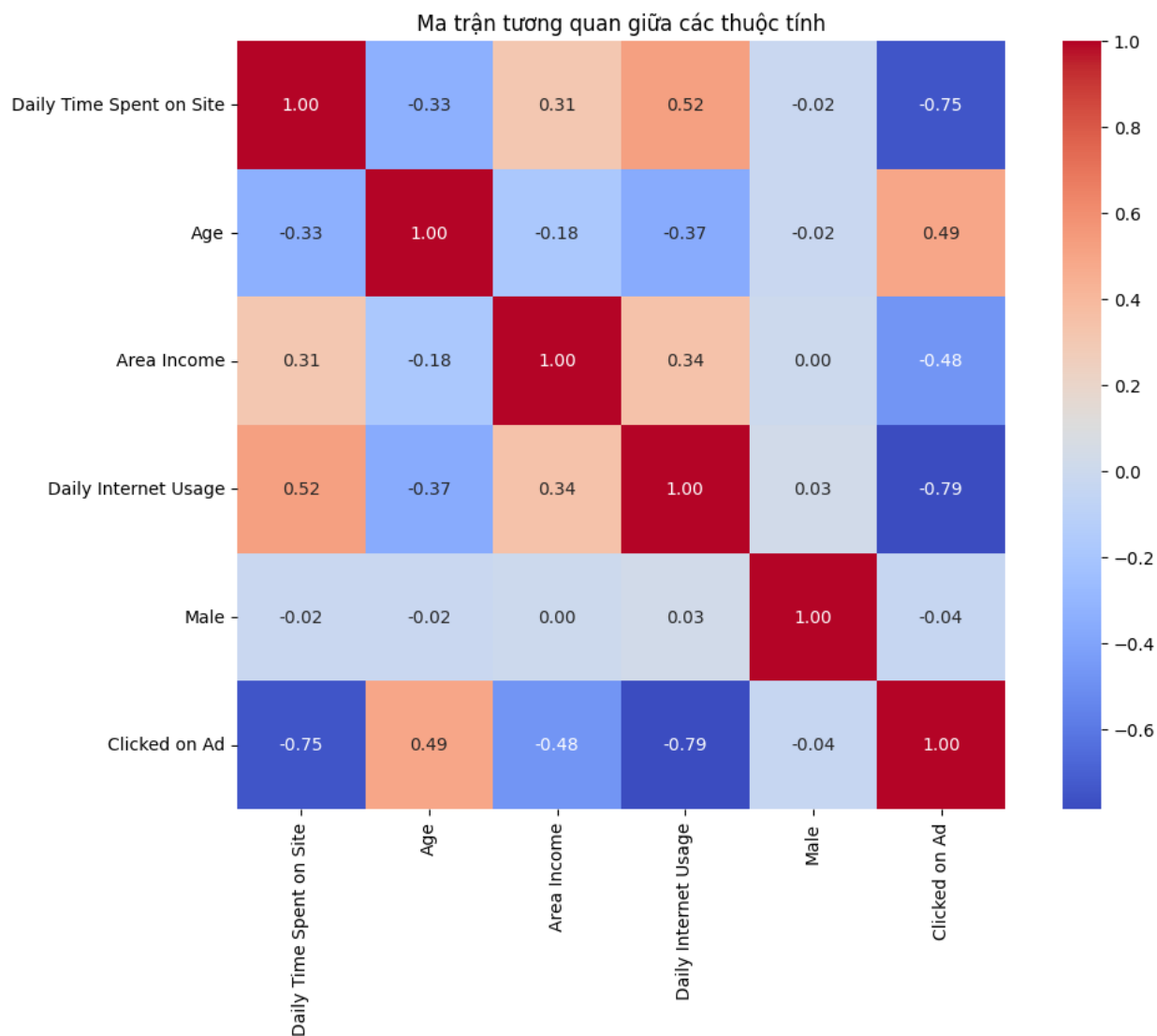


Đối với yêu cầu phân tích tần suất nhấp chuột theo giới tính, theo biểu đồ cột, có thể thấy, nam giới có xu hướng ít nhấp vào quảng cáo so với nữ giới. Tuy nhiên độ chênh lệch cũng không phải quá nhiều. Và ngược lại nữ giới sẽ nhấp quảng cáo nhiều hơn so với nam giới.



Đối với phân phân tích tần suất nhấp chuột theo tháng do nhóm có tìm hiểu thêm, dựa vào biểu đồ cột, có thể thấy tần suất nhấp chuột dao động chủ yếu trong khoảng 70-80% trong 6 tháng đầu sau đó giảm mạnh ở tháng 7. Đặc biệt, nhóm click vào quảng cáo lại tăng mạnh ở những tháng đầu năm sau đó giảm nhẹ, bắt đầu từ tháng 4.

Có thể nhận định rằng, đầu năm sẽ có xu hướng mua sắm để đổi mới nhiều hơn nên người dùng sẽ xem xét nhấp vào quảng cáo nhiều hơn.



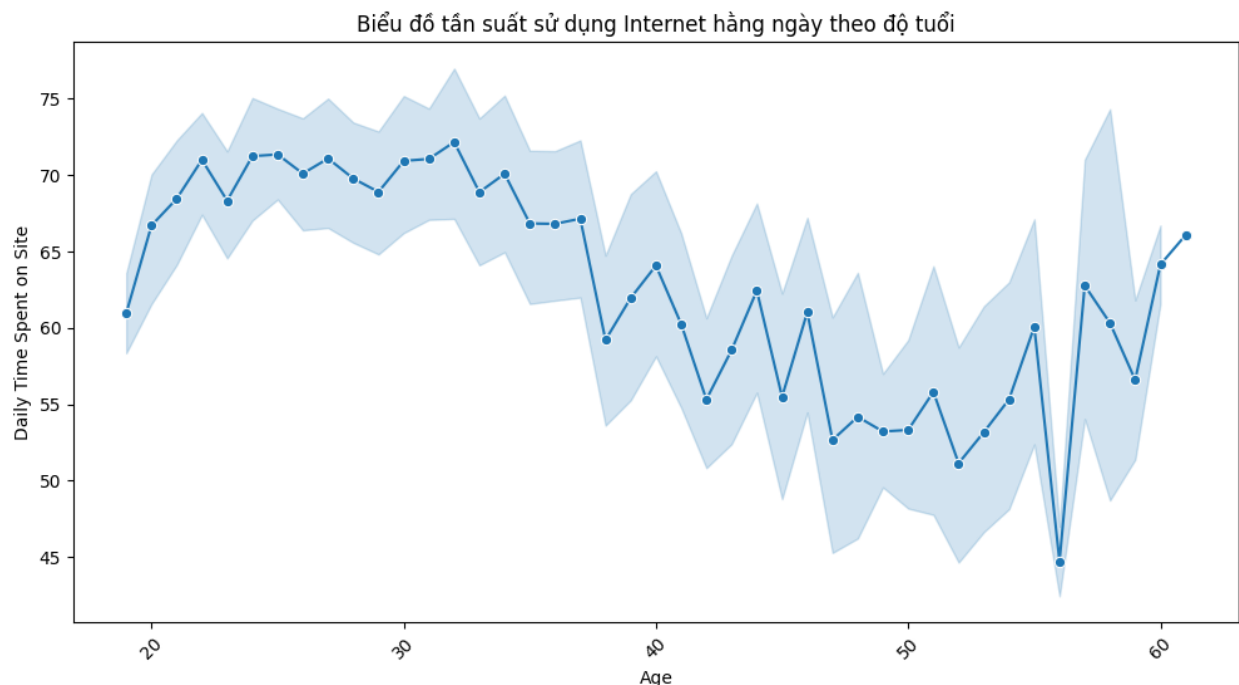
Đối với ma trận tương quan, dựa vào biểu đồ heatmap có thể nhận định rằng đối với biến Clicked on Ad, các biến như Daily Time Spent on Site, Area Income, Daily Internet Usage đều có mối tương quan nghịch và tương quan mạnh mẽ, chỉ có biến Male là độ tương quan không mạnh mẽ như các biến còn lại, thậm chí không có tương quan với Clicked on Ad.

Quan sát nhận xét về Thời gian sử dụng website theo độ tuổi, theo thu nhập hoặc theo thành phố sinh sống; Thời điểm trong ngày (sáng, trưa, chiều, tối); Chủ đề quảng cáo được quan tâm nhất hoặc ít được quan tâm nhất

Đối với phân tích và nhận xét về thời gian sử dụng website theo độ tuổi, nhóm sử dụng biểu đồ lineplot để trực quan, với trục hoành là cột Age và trục tung là cột Daily Time Spent on Site

```
plt.figure(figsize=(12, 6))
sns.lineplot(x='Age', y='Daily Time Spent on Site', data=df_unique, marker='o')
plt.title('Biểu đồ tần suất sử dụng Internet hàng ngày theo độ tuổi')
plt.xlabel('Age')
plt.ylabel('Daily Time Spent on Site')
plt.xticks(rotation=45)
plt.show()
```

Kết quả



Dựa vào biểu đồ đường, có thể nhận định rằng, xu hướng giới trẻ từ độ tuổi 20-35 có thời gian sử dụng website cao nhất với trung bình từ 70-80 phút mỗi ngày. Khi độ tuổi càng cao thì xu hướng thời gian sử dụng website sẽ giảm dần, bắt đầu từ độ tuổi 35 sẽ thấy rõ sự sụt giảm đáng kể. Đến độ tuổi từ 50-60 thì thời gian sử dụng website giảm

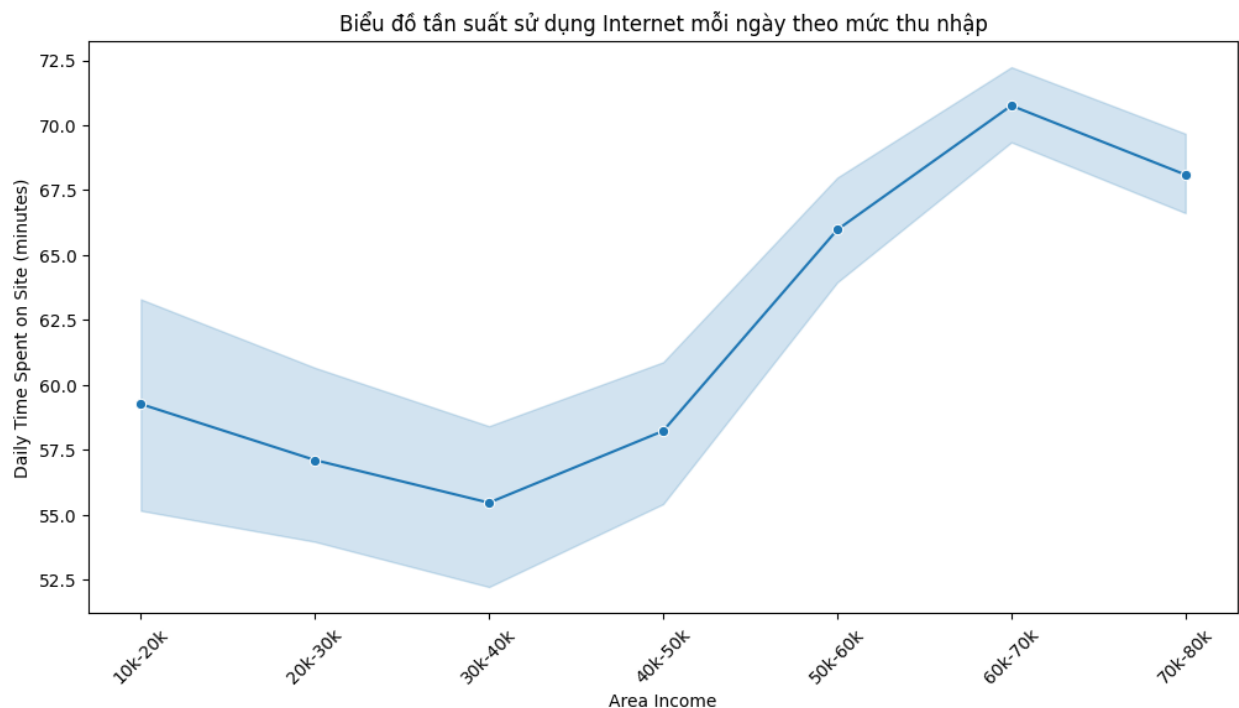
mạnh, cho thấy nhóm người trung niên từ 50-60 tuổi có thời gian sử dụng website thấp, trung bình từ 45-55 phút mỗi ngày.

Có thể kết luận rằng, giới trẻ có xu hướng truy cập website nhiều hơn, có thể do nhu cầu giải trí và công việc đòi hỏi phải truy cập nhiều hơn, trong khi nhóm người trung niên lại ít truy cập website hơn do độ tuổi càng cao thì sẽ ít có nhu cầu sử dụng công nghệ nói chung.

Đối với phân tích và nhận xét về thời gian sử dụng website theo mức thu nhập, nhóm cũng sử dụng biểu đồ lineplot để trực quan, trong đó sẽ sử dụng lại cột chỉ giá trị mức thu nhập theo từng khoảng và cũng là trục hoành, trục tung là Daily Time Spent on Site

```
plt.figure(figsize=(12, 6))
sns.lineplot(x='Income Group', y='Daily Time Spent on Site', data=df_unique, marker='o')
plt.title('Biểu đồ tần suất sử dụng Internet mỗi ngày theo mức thu nhập')
plt.xlabel('Area Income')
plt.ylabel('Daily Time Spent on Site (minutes)')
plt.xticks(rotation=45)
plt.show()
```

Kết quả



Dựa vào biểu đồ đường, có thể nhận định rằng, xu hướng những người có thu nhập thấp từ 10k-40k sẽ ít sử dụng website hơn, và mức thu nhập trong khoảng này càng

tăng thì thời gian sử dụng website cũng sẽ giảm theo, trung bình từ khoảng 55-65 phút mỗi ngày đến 55-60 phút mỗi ngày. Tuy nhiên, bắt đầu từ mức thu nhập 40k-50k thì lại có sự tăng lên rõ rệt về thời gian sử dụng website. Cụ thể từ mức thu nhập 40k đến 70k thì thời gian sử dụng website càng tăng mạnh, trung bình từ 55-65 phút mỗi ngày đến 65-75 phút mỗi ngày. Sau đó lại giảm nhẹ ở mức thu nhập từ 70k-80k.

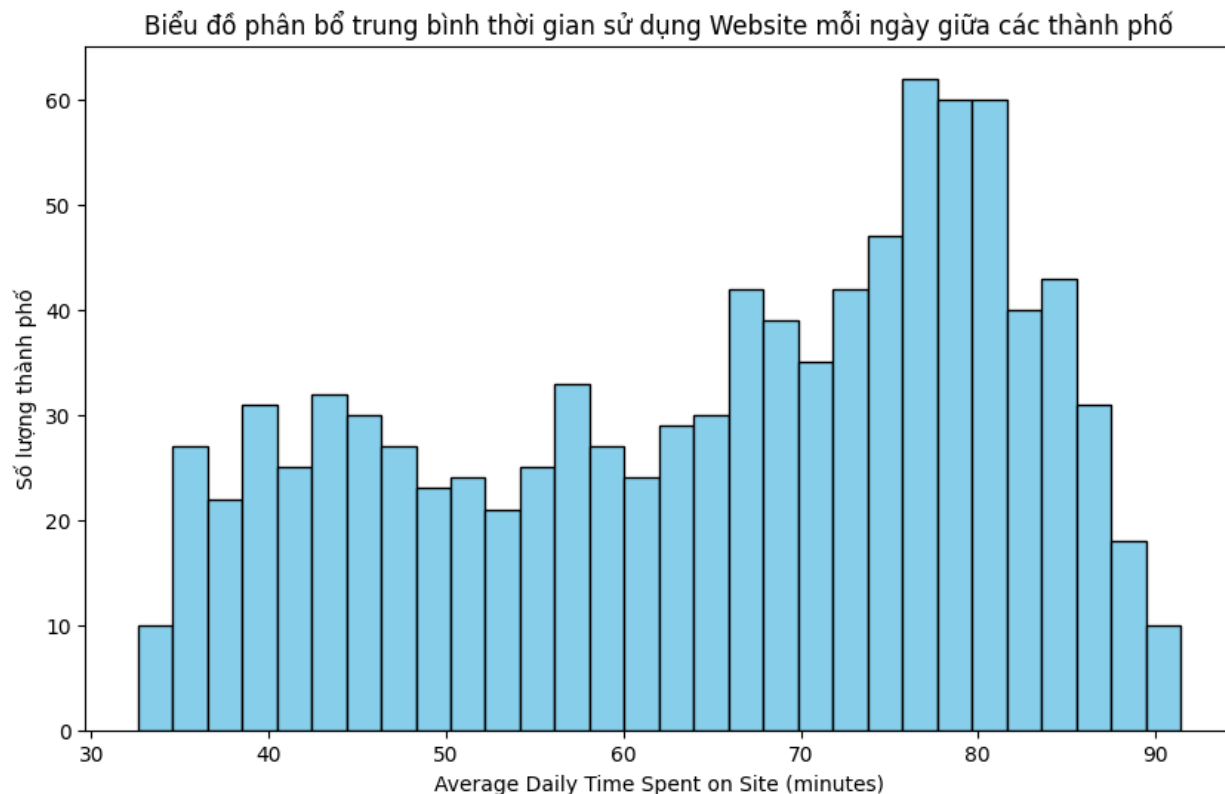
Có thể kết luận rằng, những người có mức thu nhập thấp lại ít truy cập website hơn, khi mức thu nhập tăng dần thì việc truy cập website càng tăng dần.

Đối với phân tích và nhận xét về thời gian sử dụng website theo thành phố sinh sống, do có tới hơn 900 thành phố riêng biệt nên nhóm không thể làm theo yêu cầu 1 cách hoàn chỉnh được nên nhóm có xem xét đối hướng phân tích, thành phân tích trung bình thời gian sử dụng website mỗi ngày giữa các thành phố. Nhóm sẽ lấy trung bình thời gian sử dụng website mỗi ngày và nhóm với mỗi thành phố, sau đó trực quan bằng biểu đồ histogram, với trục tung là số lượng thành phố ứng với trung bình thời gian sử dụng website mỗi ngày và trục hoành là trung bình thời gian sử dụng website mỗi ngày

```
avg_usage_by_city = df_unique.groupby('City')['Daily Time Spent on Site'].mean()

plt.figure(figsize=(10, 6))
plt.hist(avg_usage_by_city, bins=30, color='skyblue', edgecolor='black')
plt.title('Biểu đồ phân bố trung bình thời gian sử dụng Website mỗi ngày giữa các thành phố')
plt.xlabel('Average Daily Time Spent on Site (minutes)')
plt.ylabel('Number of Cities')
plt.show()
```

Kết quả



Dựa vào biểu đồ histogram, có thể nhận định, đa số các thành phố có trung bình thời gian sử dụng website mỗi ngày dao động từ 60 đến 80 phút mỗi ngày, và gần như rất ít thành phố có trung bình thời gian sử dụng website dưới 30 phút mỗi ngày, cũng như rất ít thành phố có trung bình thời gian sử dụng website trên 90 phút mỗi ngày.

Có thể kết luận, phần lớn nhiều thành phố có người dân có thời gian sử dụng website từ 1h đến 1h30p mỗi ngày.

Ngoài ra để bổ sung thêm cho phần phân tích thời gian sử dụng website theo thành phố sinh sống, nhóm có làm thêm về phân tích top 10 thành phố có thời gian sử dụng website mỗi ngày nhiều nhất, và sẽ sử dụng biểu đồ cột để trực quan hóa, với trục hoành là top 10 thành phố và trục tung là Daily Time Spent on Site. Để làm được phân tích này thì trước tiên sẽ tính trung bình thời gian sử dụng website mỗi ngày và nhóm vào các thành phố, sau đó sẽ lấy ra top 10 giá trị cao nhất và dùng biểu đồ cột để trực quan.

```

avg_usage_by_city = df_unique.groupby('City')['Daily Time Spent on Site'].mean()

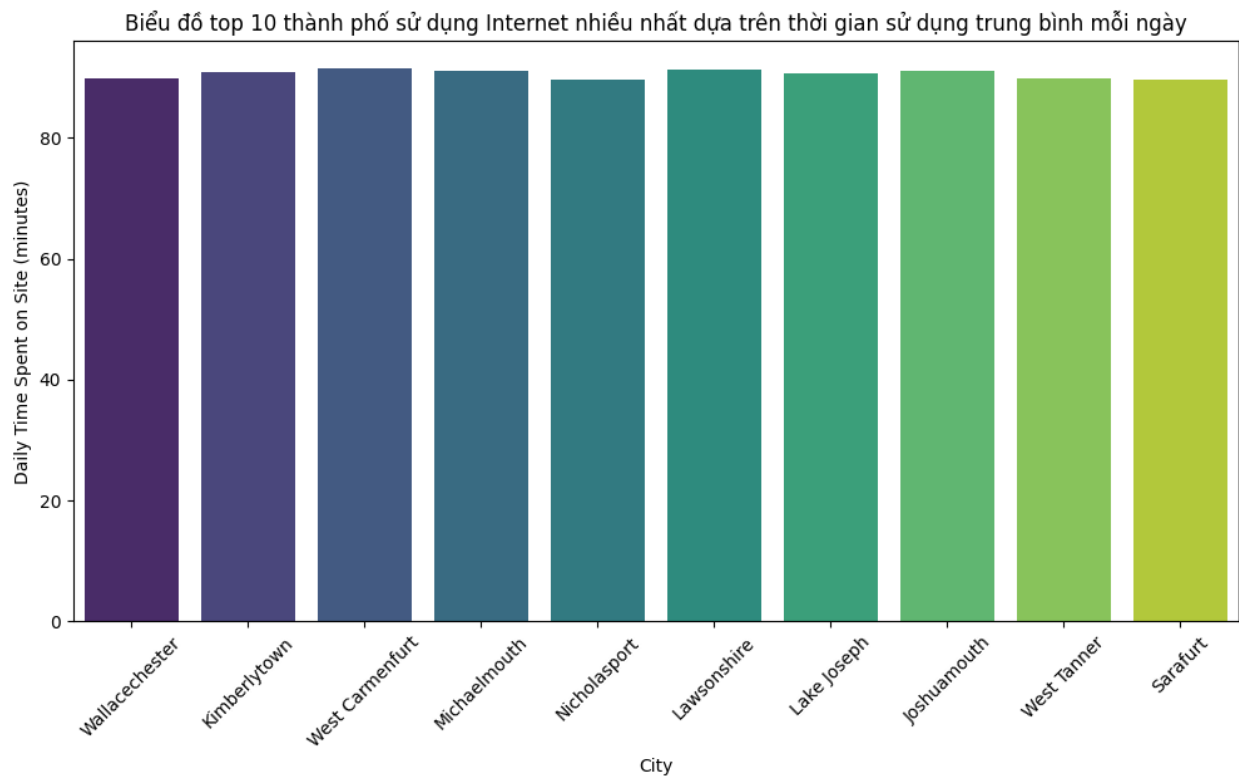
top_10_cities = avg_usage_by_city.nlargest(10).index

filtered_df = df_unique[df_unique['City'].isin(top_10_cities)]

plt.figure(figsize=(12, 6))
sns.barplot(x='City', y='Daily Time Spent on Site', data=filtered_df, palette='viridis', ci=None)
plt.title('Biểu đồ top 10 thành phố sử dụng Internet nhiều nhất dựa trên thời gian sử dụng trung bình mỗi ngày')
plt.xlabel('City')
plt.ylabel('Daily Time Spent on Site (minutes)')
plt.xticks(rotation=45)
plt.show()

```

Kết quả



Dựa vào biểu đồ cột, có thể nhận định, có 3 thành phố có người dân có thời gian sử dụng website cao nhất lần lượt là West Carmenfurt, Lawsonshire và Joshuamouth với thời gian sử dụng gần 90 phút mỗi ngày. Tiếp theo là các thành phố Kimberllytown và Michaelmouth với thời gian sử dụng ít hơn nhưng không đáng kể, và các thành phố còn lại.

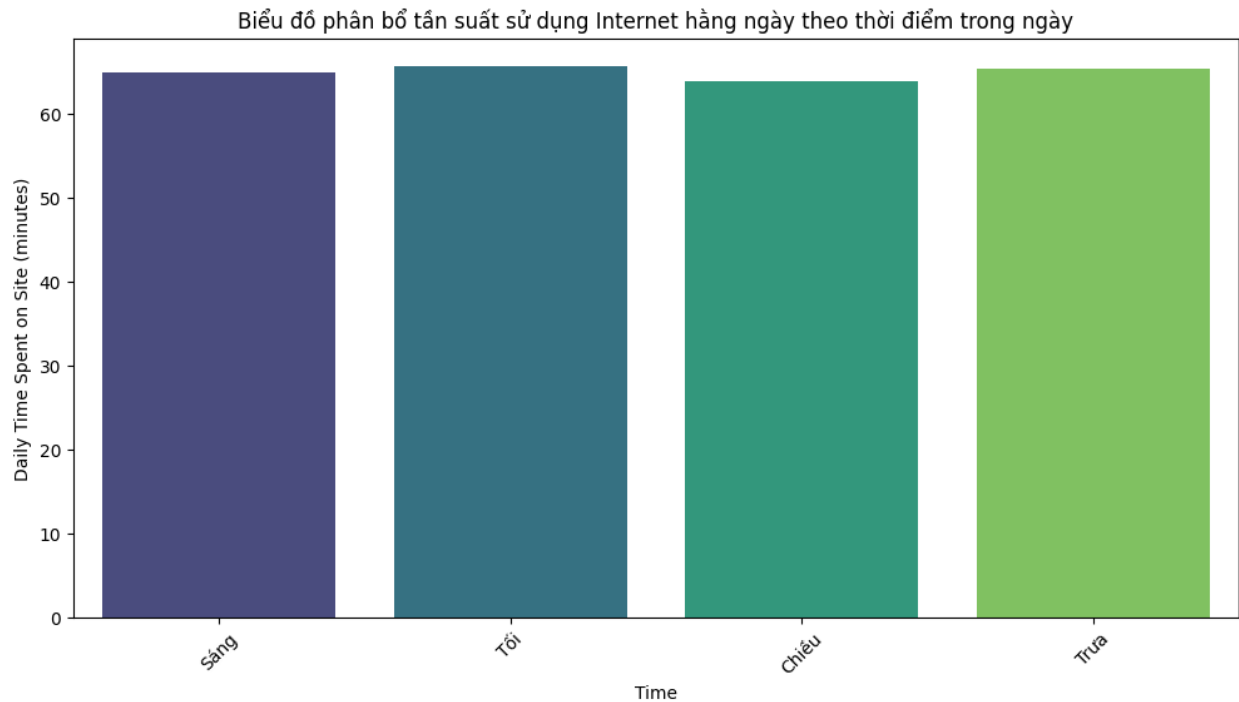
Đối với phân tích nhận xét về thời gian sử dụng website trong ngày, trước tiên nhóm sẽ dùng hàm `time_group` với `hour` là biến đầu vào, hàm này có chức năng sẽ rút trích dữ liệu và dùng `if else` để xác định, nếu giá trị từ 0 đến 10 sẽ trả về “Sáng”, từ 10 đến 13 trả về “Trưa”, từ 13 đến 18 trả về “Chiều”, còn lại trả về “Tối”. Lý do sử dụng hàm này là ở giai đoạn khai phá dữ liệu, nhóm có tách thuộc tính `Timestamp` thành 2 thuộc tính là `Date` và `Time`, nhóm sẽ sử dụng thuộc tính `Time` (đại diện cho thời gian) và áp dụng hàm `time_group`. Trước tiên sẽ biến đổi kiểu dữ liệu của thuộc tính `Time` thành `datetime`, sau đó sẽ tạo thêm cột `Time Group` (đại diện cho thời gian trong ngày) và áp dụng hàm `time_group` vào. Cuối cùng là sử dụng thuộc tính `Time Group` vừa tạo làm trục hoành và `Daily Time Spent on Site` làm trục tung cho biểu đồ cột và trực quan.

```
def time_group(hour):
    if 0 <= hour < 10:
        return 'Sáng'
    elif 10 <= hour < 13:
        return 'Trưa'
    elif 13 <= hour < 18:
        return 'Chiều'
    else:
        return 'Tối'

df_unique['Hour'] = pd.to_datetime(df_unique['Time'], format='%H:%M').dt.hour
df_unique['Time Group'] = df_unique['Hour'].apply(time_group)
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x='Time Group', y='Daily Time Spent on Site', data=df_unique, palette='viridis', ci=None)
plt.title('Biểu đồ phân bố tần suất sử dụng Internet hàng ngày theo thời điểm trong ngày')
plt.xlabel('Time')
plt.ylabel('Daily Time Spent on Site (minutes)')
plt.xticks(rotation=45)
plt.show()
```

Kết quả



Dựa vào biểu đồ cột, có thể nhận định, khoảng thời gian “Tối” và “Trưa” là 2 khoảng thời gian có tần suất sử dụng website mỗi ngày cao nhất với hơn 1h. Có thể do đây là 2 khoảng thời gian mà đa số người dân có thời gian rảnh sẽ sử dụng website. 2 khoảng thời gian còn lại chỉ thấp hơn không đáng kể so với khoảng thời gian “Tối” và “Trưa”.

Đối với phân tích nhận xét chủ đề quảng cáo được quan tâm nhất, nhóm sẽ chọn ra top 10 chủ đề quảng cáo dựa trên thời gian sử dụng website cao nhất. Cách làm sẽ tương tự như lấy top 10 thành phố dựa trên thời gian sử dụng website.

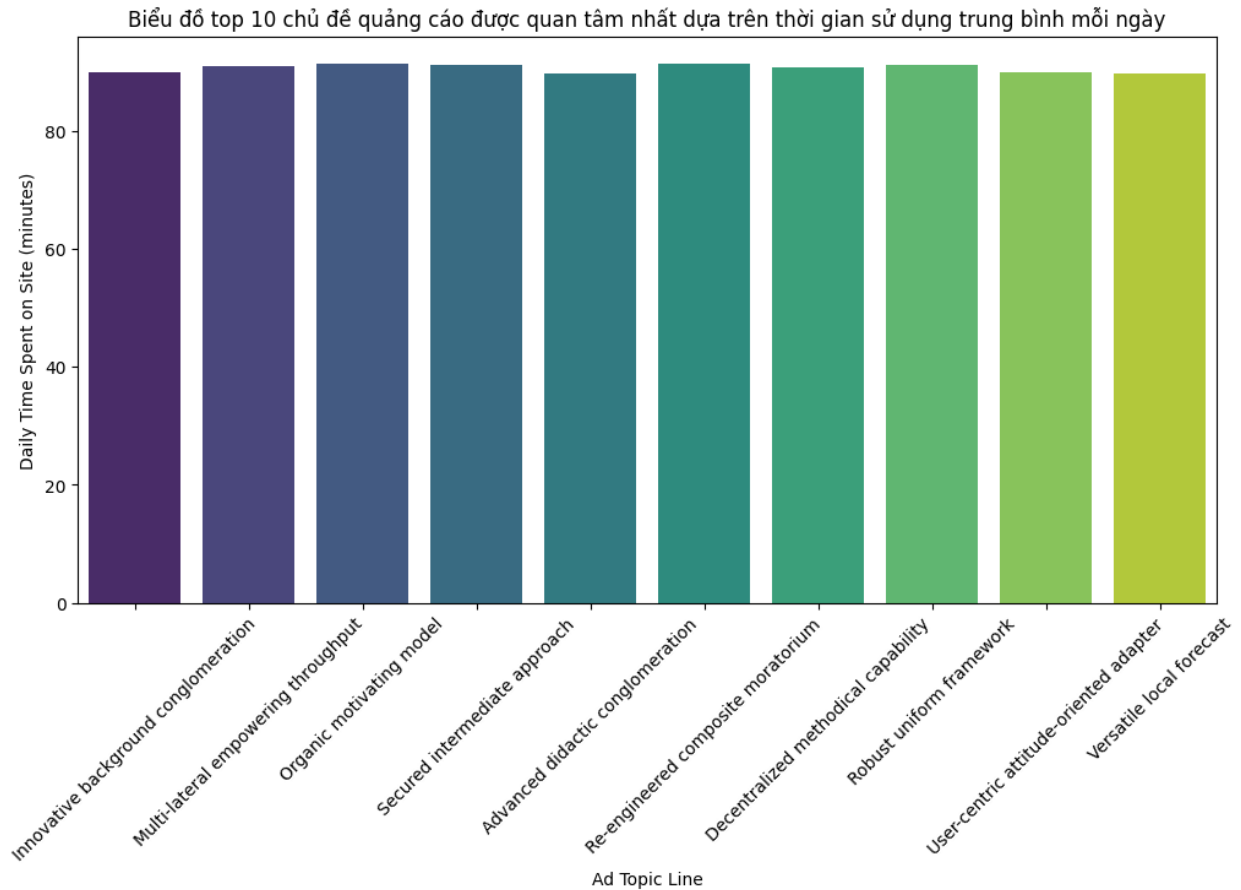
```
avg_usage_by_topic = df_unique.groupby('Ad Topic Line')['Daily Time Spent on Site'].mean()

top_10_topics = avg_usage_by_topic.nlargest(10).index

filtered_df = df_unique[df_unique['Ad Topic Line'].isin(top_10_topics)]

plt.figure(figsize=(12, 6))
sns.barplot(x='Ad Topic Line', y='Daily Time Spent on Site', data=filtered_df, palette='viridis', ci=None)
plt.title('Biểu đồ top 10 chủ đề quảng cáo được quan tâm nhất dựa trên thời gian sử dụng trung bình mỗi ngày')
plt.xlabel('Ad Topic Line')
plt.ylabel('Daily Time Spent on Site (minutes)')
plt.xticks(rotation=45)
plt.show()
```

Kết quả



Dựa vào biểu đồ cột, có thể nhận định, chủ đề "Organic motivating model", "Re-engineered composite moratorium" được quan tâm nhất với thời gian sử dụng website từ 85-90 phút, các chủ đề còn lại có mức thời gian khá đồng đều, từ 80 phút trở lên, và sự chênh lệch giữa các chủ đề không quá cao.

Dự đoán khả năng 1 người dùng có chọn xem quảng cáo được hiển thị hay không (lựa chọn và cài đặt thuật toán, cài đặt ít nhất 2 thuật toán để so sánh với nhau)

Để kiểm tra sự ảnh hưởng của thuộc tính thu nhập cũng như các thuộc tính khác có ảnh hưởng đến mô hình dự đoán hay không, trước tiên sẽ lọc và xóa bớt những thuộc tính không cần thiết cũng như không ảnh hưởng đến việc dự đoán. Đầu tiên sẽ đẩy thuộc tính Clicked on Ad về cuối bảng do trong quá trình phân tích có xuất hiện thêm nhiều thuộc tính mới để hỗ trợ trong việc phân tích.

```
df_unique = df_unique[[col for col in df_unique.columns if col != 'Clicked on Ad']] + [['Clicked on Ad']]
df_unique
```

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Date	Time	Income Group	Month	Hour	Time Group	Clicked on Ad
0	68.95	35.0	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	27/3/2016 0:53	2016-03-27	0:53	60k-70k	3	0	Sáng	0
1	80.23	31.0	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	4/4/2016 1:39	2016-04-04	1:39	60k-70k	4	1	Sáng	0
2	69.47	26.0	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	13/3/2016 20:35	2016-03-13	20:35	50k-60k	3	20	Tối	0
3	74.15	29.0	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	10/1/2016 2:31	2016-01-10	2:31	50k-60k	1	2	Sáng	0
4	68.37	35.0	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	3/6/2016 3:36	2016-06-03	3:36	70k-80k	6	3	Sáng	0
...
996	72.97	30.0	71384.57	208.58	Fundamental modular algorithm	Duffystad	1	Lebanon	11/2/2016 21:49	2016-02-11	21:49	70k-80k	2	21	Tối	1
997	51.30	45.0	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	1	Bosnia and Herzegovina	22/4/2016 2:07	2016-04-22	2:07	60k-70k	4	2	Sáng	1
998	51.63	51.0	42415.72	120.37	Expanded intangible solution	South Jessica	1	Mongolia	1/2/2016 17:24	2016-02-01	17:24	40k-50k	2	17	Chiều	1
999	55.55	19.0	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	0	Guatemala	24/3/2016 2:35	2016-03-24	2:35	40k-50k	3	2	Sáng	0
1000	45.01	35.0	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	0	Brazil	3/6/2016 21:43	2016-06-03	21:43	20k-30k	6	21	Tối	1

Sau đó sẽ xem xét nên xóa bớt thuộc tính nào mà không cần thiết và không ảnh hưởng đến việc dự đoán. Ở đây nhóm nhận thấy chỉ có những thuộc tính trong ma trận tương quan vừa này là cần thiết và có ảnh hưởng đến mô hình dự đoán, nên những thuộc tính còn lại sẽ bị xóa.

```
df_unique.drop(['Ad Topic Line', 'City', 'Country', 'Income Group', 'Time Group', 'Hour', 'Date', 'Month', 'Time', 'Timestamp'], axis=1, inplace=True)
df_unique
```

<ipython-input-39-f13591c144fb>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df_unique.drop(['Ad Topic Line', 'City', 'Country', 'Income Group', 'Time Group', 'Hour', 'Date', 'Month', 'Time', 'Timestamp'], axis=1, inplace=True)
```

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
0	68.95	35.0	61833.90	256.09	0	0
1	80.23	31.0	68441.85	193.77	1	0
2	69.47	26.0	59785.94	236.50	0	0
3	74.15	29.0	54806.18	245.89	1	0
4	68.37	35.0	73889.99	225.58	0	0
...
996	72.97	30.0	71384.57	208.58	1	1
997	61.30	45.0	67782.17	134.42	1	1
998	61.63	51.0	42415.72	120.37	1	1
999	65.55	19.0	41920.79	187.95	0	0
1000	45.01	35.0	29875.80	178.35	0	1

1000 rows × 6 columns

Sau đó tiến hành chia tập dữ liệu, với x là 5 thuộc tính độc lập và y là thuộc tính phụ thuộc (tức Clicked on Ad). Tiếp theo sẽ tiến hành train/test tập dữ liệu với tỷ lệ 1/3 cho tập test và 2/3 cho tập train. Sau đó sử dụng hàm StandardScaler() để chuẩn hóa x_train và x_test, mục đích để làm cho độ lệch chuẩn bằng 1, từ đó các mô hình Machine Learning sẽ học và dự đoán hiệu quả hơn.

```
x1 = df_unique.iloc[:, :-1]
y1 = df_unique.iloc[:, -1]

x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=1/3, random_state=0)

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

x_train_with_const = sm.add_constant(x_train_scaled)

model = sm.Logit(y_train, x_train_with_const)
result = model.fit()

print(result.summary())
```

Sau đó sẽ sử dụng hàm add_constant để thêm hằng số intercept vào x_train và sử dụng hàm Logit từ thư viện statsmodel. Tương tự như OLS, Logit sẽ tạo mô hình Logistic Regression với biến mục tiêu là y_train và đầu vào là x_train_with_const để phân tích mối tương quan giữa đầu vào và biến mục tiêu. Sau đó sẽ huấn luyện và xuất ra kết quả.

Kết quả

Optimization terminated successfully.
Current function value: 0.093830
Iterations 10

Logit Regression Results

Dep. Variable:	Clicked on Ad	No. Observations:	666
Model:	Logit	Df Residuals:	660
Method:	MLE	Df Model:	5
Date:	Tue, 17 Dec 2024	Pseudo R-squ.:	0.8645
Time:	09:16:59	Log-Likelihood:	-62.490
converged:	True	LL-Null:	-461.05
Covariance Type:	nonrobust	LLR p-value:	4.872e-170

	coef	std err	z	P> z	[0.025	0.975]
const	2.2648	0.438	5.176	0.000	1.407	3.122
x1	-3.1199	0.411	-7.592	0.000	-3.925	-2.315
x2	1.1689	0.254	4.598	0.000	0.671	1.667
x3	-2.0944	0.358	-5.846	0.000	-2.797	-1.392
x4	-2.8090	0.363	-7.744	0.000	-3.520	-2.098
x5	-0.0787	0.243	-0.324	0.746	-0.555	0.398

Dựa vào kết quả trên, có thể nhận định 1 vài chi tiết như sau :

- Const với giá trị 2.265, cho thấy khi tất cả các biến đặc trưng bằng 0 thì mô hình Logistic Regression sẽ tính toán xác suất nhấp quảng cáo bằng hệ số này. Do hệ số này lớn nên mô hình tính toán khả năng nhấp quảng cáo là khá cao. Các biến độc lập thay đổi như thế nào thì sẽ làm tăng hoặc giảm khả năng nhấp quảng cáo. Cộng với p-value rất nhỏ (chạm ngưỡng 0), hệ số này có ý nghĩa thống kê, dẫn đến có ảnh hưởng đáng kể tới mô hình.
- x1 (ứng với Daily Time Spent on Site), với hệ số bằng -3.112, cho thấy có mối tương quan nghịch biến với khả năng nhấp quảng cáo, dẫn đến có khả năng làm giảm xác suất nhấp quảng cáo. Cùng với p-value cực nhỏ (chạm ngưỡng 0), x1 có ý nghĩa thống kê, dẫn đến có ảnh hưởng đến mô hình.
- x2 (ứng với Age), với hệ số bằng 1.17, cho thấy có mối tương quan đồng biến với khả năng nhấp quảng cáo, dẫn đến làm tăng xác suất nhấp quảng cáo. Cùng với p-value cực nhỏ (chạm ngưỡng 0), x2 cũng có ý nghĩa thống kê, dẫn đến có ảnh hưởng đến mô hình.
- x3 (ứng với Area Income), với hệ số bằng -2.094, cho thấy có mối tương quan nghịch biến với khả năng nhấp quảng cáo, dẫn đến có khả năng làm giảm xác suất nhấp quảng cáo. Cùng với p-value cực nhỏ (chạm ngưỡng 0), x3 cũng có ý nghĩa thống kê, dẫn đến có ảnh hưởng đến mô hình.
- x4 (ứng với Daily Internet Usage), với hệ số bằng -2.8090, cho thấy có mối tương quan nghịch biến với khả năng nhấp quảng cáo, dẫn đến có khả năng làm giảm xác suất nhấp quảng cáo. Cùng với p-value cực nhỏ (chạm ngưỡng 0), x4 cũng có ý nghĩa thống kê, dẫn đến có ảnh hưởng đến mô hình.

- x5 (ứng với Male), với hệ số bằng -0.0787, vì hệ số gần bằng 0 nên có thể nhận định Male không có mối tương quan rõ rệt với xác suất nhấp quảng cáo. Cộng với p-value bằng 0.746, khi p-value lớn thì đồng nghĩa với việc ít có ý nghĩa thống kê nên có thể kết luận biến Male không ảnh hưởng đến mô hình dự đoán xác suất nhấp quảng cáo.

Có thể kết luận rằng, Daily Time Spent on Site, Age, Area Income, Daily Internet Usage đều có ảnh hưởng đến mô hình dự đoán xác suất nhấp quảng cáo. Trong đó Age có mối tương quan đồng biến, còn lại là mối tương quan nghịch biến.

Tiếp tục đến phần cài đặt thuật toán, nhóm quyết định chọn 2 mô hình thuật toán là Logistic Regression và K-Nearest Neighbors.

Đầu tiên, đối với thuật toán Logistic Regression, trước tiên vẫn giữ nguyên tập train/test. Sau đó sẽ tạo 1 mô hình Logistic Regression từ chính thư viện của Logistic Regression với các tham số solver là liblinear (Giải thuật tối ưu hóa dựa trên phép lập trình tuyến tính phù hợp cho bài toán hồi quy logistic nhị phân và dữ liệu nhỏ, ở đây là 1000 dòng dữ liệu) và max_iter là 1000 (Số vòng lặp tối đa để thuật toán tối ưu hóa hội tụ, đảm bảo huấn luyện mô hình đầy đủ.). Sau đó sẽ tiến hành huấn luyện mô hình với x_train và y_train. Tiếp đó là dự đoán trên x_test và đánh giá mô hình bằng accuracy_score, đồng thời hiển thị báo cáo phân loại trên tập test cũng như ma trận nhầm lẫn trên tập test (được trực quan bằng biểu đồ heatmap).

```
# Khởi tạo mô hình Logistic Regression với tham số solver='liblinear' và max_iter=1000
regression = LogisticRegression(solver='liblinear', max_iter=1000)

# Huấn luyện mô hình với dữ liệu huấn luyện
regression.fit(x_train_scaled, y_train)

# Dự đoán trên dữ liệu kiểm tra
y_pred = regression.predict(x_test_scaled)

# Đánh giá mô hình trên tập kiểm tra
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Set: {accuracy:.4f}")

# Hiển thị báo cáo phân loại (precision, recall, f1-score) trên tập kiểm tra
print("Classification Report on Test Set:")
print(classification_report(y_test, y_pred))

# Ma trận nhầm lẫn trên tập kiểm tra
print("Confusion Matrix on Test Set:")
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

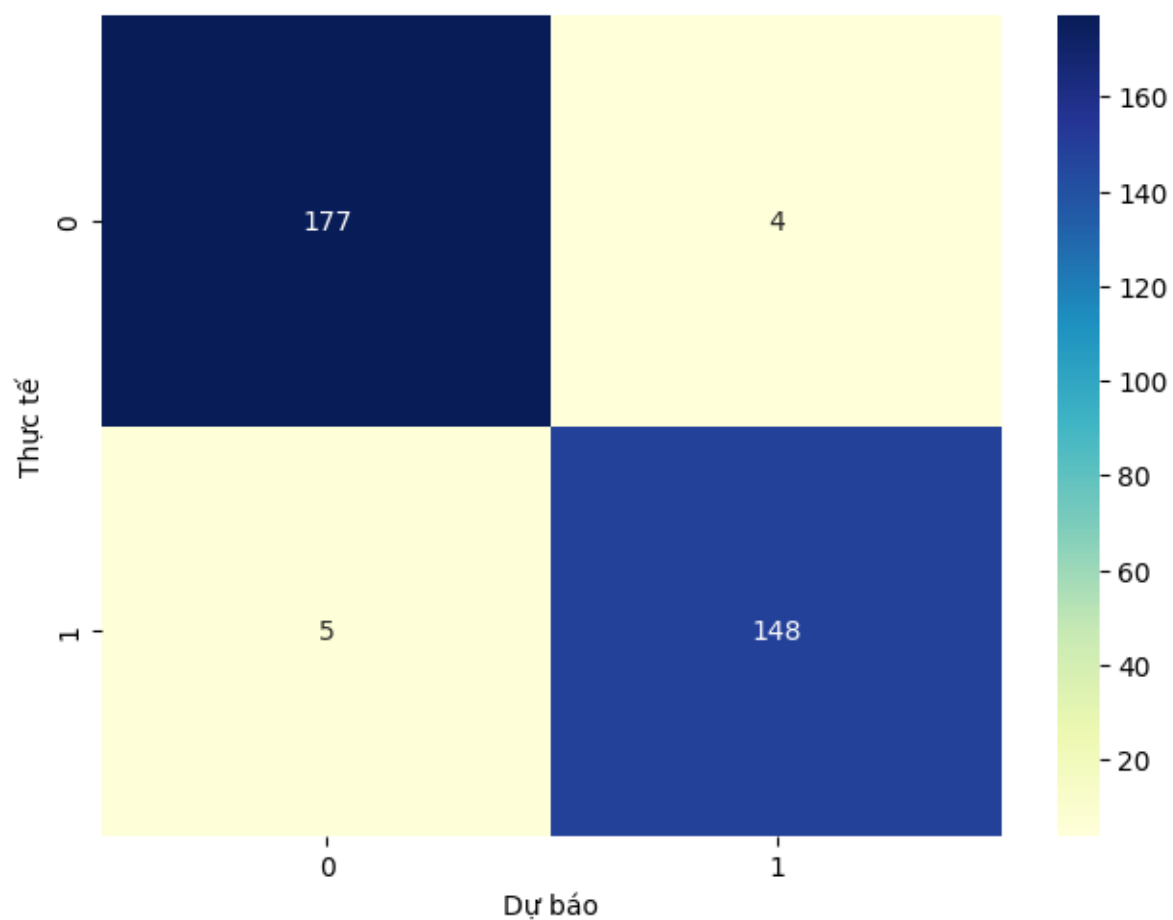
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu", fmt='g')
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Thực tế')
plt.xlabel('Dự báo')
```

Kết quả

```
Accuracy on Test Set: 0.9731
Classification Report on Test Set:
```

	precision	recall	f1-score	support
0	0.97	0.98	0.98	181
1	0.97	0.97	0.97	153
accuracy			0.97	334
macro avg	0.97	0.97	0.97	334
weighted avg	0.97	0.97	0.97	334

Confusion matrix



Dựa trên kết quả có được, có 1 vài nhận định như sau :

- Độ Accuracy trên tập test là 97.31%, cho thấy mô hình trên tập test khá ổn, có thể dự đoán chính xác hầu hết tập dữ liệu.
- Độ Precision (tức tỷ lệ giữa số sample được tính là True Positive (TP) với tổng số sample được phân loại là Positive (bằng chính TP + FP)) của hai lớp 0 và 1 đều là 97%, cho thấy các mẫu dữ liệu khi dự đoán các lớp 0 và 1, 97% sẽ đúng.
- Độ Recall (tỷ lệ giữa các điểm True Positive được nhận đúng trên tổng điểm True Positive) của lớp 0 là 98%, của lớp 1 là 97%, cho thấy các mẫu dữ liệu thực tế là lớp 0, 98% được dự đoán hoàn toàn đúng, còn mẫu dữ liệu thực tế là lớp 1, 97% được dự đoán hoàn toàn đúng.
- Độ F1-Score của cả 2 lớp đều là 97-98%, cho thấy mô hình hoạt động ổn định trên cả 2 lớp, không có sự mất cân bằng trong việc dự đoán dữ liệu
- Về Macro Average và Weighted Average, cả 2 đều bằng 97%, cho thấy mô hình hoạt động đồng đều trên cả hai lớp và không có sự thiên vị về một lớp cụ thể nào. Cộng với giá trị Support của lớp 0 và lớp 1 lần lượt là 181 và 153, cho thấy không xảy ra hiện tượng class imbalance.

Với mô hình thứ 2 là K-Nearest Neighbors, đầu tiên quy trình vẫn sẽ là chia tập train/test theo tỷ lệ 1/3 cho dữ liệu test, 2/3 cho dữ liệu train, sau đó chuẩn hóa dữ liệu x_train và x_test để giúp mô hình học tốt hơn, dự đoán tốt hơn. Kế tiếp là tạo mô hình K-Nearest Neighbors với tham số ban đầu mặc định n-neighbors = 3, tức là lấy 1 điểm dữ liệu bất kỳ trong tập test, sau đó tính khoảng cách từ điểm dữ liệu của tập test đến tất cả các điểm dữ liệu trong tập train, và chọn ra 3 điểm có khoảng cách gần nhất so với điểm dữ liệu trên tập test. Sau đó thuật toán sẽ xem xét nhãn của 3 điểm gần nhất, nhãn nào xuất hiện nhiều hơn thì sẽ gán nhãn đó cho điểm dữ liệu trên tập test, và quá trình này sẽ kết thúc khi tất cả các điểm dữ liệu trong tập test được dự đoán xong.

```
#Chia dữ liệu thành tập huấn luyện và kiểm tra
x_train, x_test, y_train, y_test = train_test_split(x1, y1, test_size=1/3, random_state=0)

#Chuẩn hóa dữ liệu để đảm bảo các đặc trưng có cùng thang đo
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

#Huấn luyện mô hình KNN ban đầu với n_neighbors=3
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(x_train_scaled, y_train)
```

Sau đó sẽ tính accuracy_score cho mô hình, cùng báo cáo phân loại và ma trận nhầm lẫn

```

print("\nĐánh giá mô hình ban đầu:")
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Set: {accuracy:.4f}")
print("Classification Report on Test Set:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix on Test Set:")
print(confusion_matrix(y_test, y_pred))

```

Kết quả

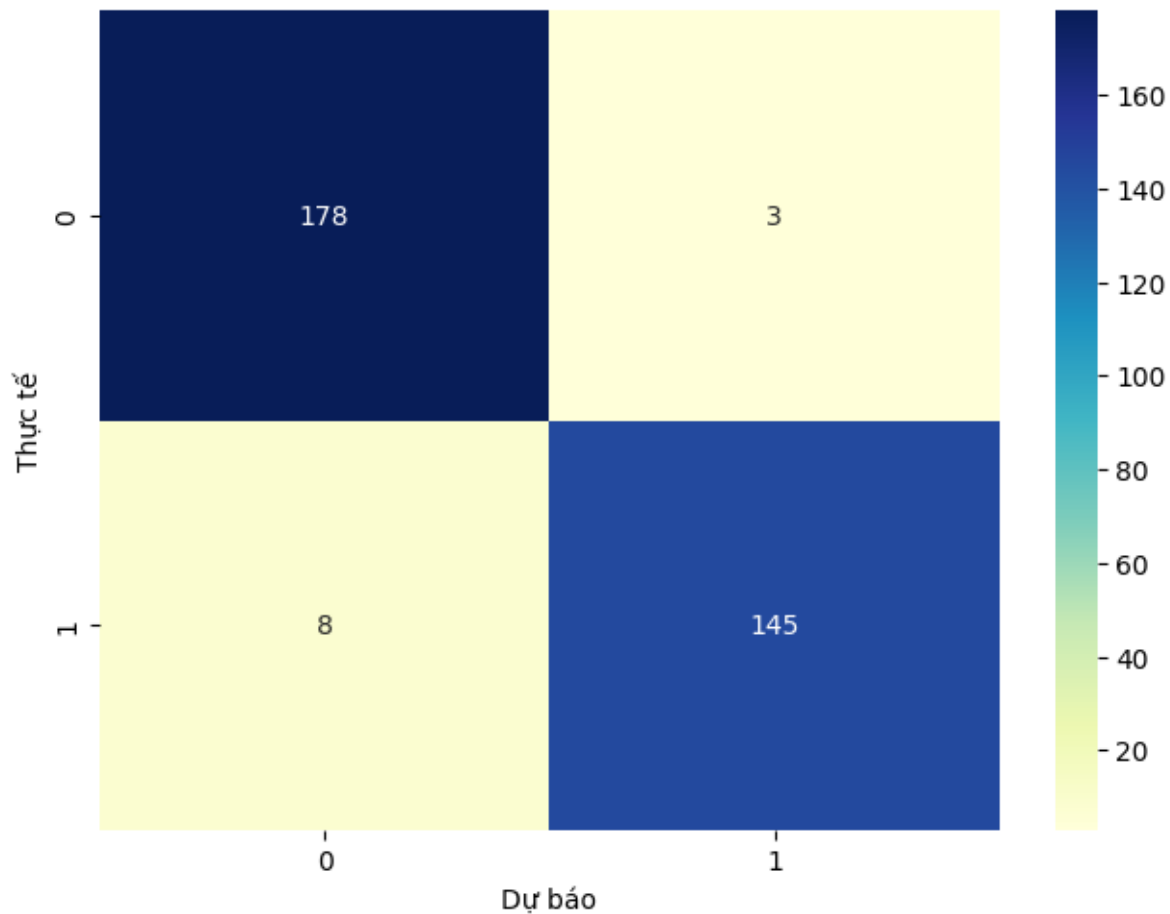
```

Đánh giá mô hình ban đầu:
Accuracy on Test Set: 0.9671
Classification Report on Test Set:

```

	precision	recall	f1-score	support
0	0.96	0.98	0.97	181
1	0.98	0.95	0.96	153
accuracy			0.97	334
macro avg	0.97	0.97	0.97	334
weighted avg	0.97	0.97	0.97	334

Confusion matrix



Dựa trên kết quả có được, có 1 vài nhận định như sau :

- Độ Accuracy trên tập test là 96.71%, cho thấy mô hình trên tập test khá ổn, có thể dự đoán chính xác hầu hết tập dữ liệu.
- Độ Precision (tỷ lệ giữa số sample được tính là True Positive (TP) với tổng số sample được phân loại là Positive (bằng chính TP + FP)) của hai lớp 0 và 1 lần lượt là 96% và 98%, cho thấy các mẫu dữ liệu khi dự đoán các lớp 0 và 1, 96-98% sẽ đúng.
- Độ Recall (tỷ lệ giữa các điểm True Positive được nhận đúng trên tổng điểm True Positive) của lớp 0 là 98%, của lớp 1 là 95%, cho thấy các mẫu dữ liệu thực tế là lớp 0, 98% được dự đoán hoàn toàn đúng, còn mẫu dữ liệu thực tế là lớp 1, 95% được dự đoán hoàn toàn đúng.
- Độ F1-Score của cả 2 lớp đều là 96-97%, cho thấy mô hình hoạt động ổn định trên cả 2 lớp, không có sự mất cân bằng trong việc dự đoán dữ liệu
- Về Macro Average và Weighted Average, cả 2 đều bằng 97%, cho thấy mô hình hoạt động đồng đều trên cả hai lớp và không có sự thiên vị về một lớp cụ thể

nào. Cộng với giá trị Support của lớp 0 và lớp 1 lần lượt là 181 và 153, cho thấy không xảy ra hiện tượng class imbalance.

Tuy nhiên bản chất của thuật toán K-Nearest Neighbors là KNN nhạy với dữ liệu bị nhiễu hoặc outliers nên với n-neighbors càng nhỏ thì mô hình sẽ càng nhạy cảm với dữ liệu bị nhiễu dẫn đến dự đoán có thể có sai sót và mô hình không ổn định. Do đó cần phải tăng n-neighbors tức tăng số lượng điểm có khoảng cách gần nhất từ tất cả các điểm dữ liệu trên tập train đến điểm dữ liệu trên tập test để chọn ra bao nhiêu điểm sẽ phù hợp với mô hình và đưa ra dự đoán chính xác nhất có thể.

Để làm được điều đó, nhóm sẽ dùng Grid Search để tìm n-neighbors phù hợp nhất với mô hình, sau đó sẽ huấn luyện lại mô hình với n-neighbors phù hợp nhất. Trước tiên sẽ tạo 1 grid từ 1 đến 20, tương ứng với phạm vi của n-neighbors từ 1 đến 20. Sau đó tạo 1 object GridSearchCV với mô hình KNN cần tối ưu, sử dụng kết hợp với 5-fold cross-validation và lấy giá trị Accuracy làm tham chiếu đánh giá. Sau đó sẽ huấn luyện và thử tất cả giá trị n-neighbors từ 1 đến 20. Sau khi huấn luyện xong, Grid Search sẽ đưa ra giá trị n-neighbors phù hợp nhất dựa trên kết quả phân loại tốt nhất (tức accuracy cao nhất) trên tập validation. Cuối cùng là sử dụng giá trị n-neighbors đã được tối ưu và huấn luyện lại mô hình KNN 1 lần nữa.

```
#Tìm giá trị tối ưu của k bằng Grid Search
param_grid = {'n_neighbors': range(1, 21)}
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, scoring='accuracy')
grid_search.fit(x_train_scaled, y_train)

best_k = grid_search.best_params_['n_neighbors']
print(f"\nK tối ưu nhất cho mô hình : {best_k}")

#Huấn luyện lại với k tối ưu
optimal_knn = KNeighborsClassifier(n_neighbors=best_k)
optimal_knn.fit(x_train_scaled, y_train)
```

Sau đó đưa ra dự đoán và đánh giá mới trên tập kiểm tra, đồng thời tính accuracy_score cho mô hình, cùng báo cáo phân loại và ma trận nhầm lẫn mới


```

#Dự đoán và đánh giá trên tập kiểm tra
y_pred_optimal = optimal_knn.predict(x_test_scaled)

print("\nĐánh giá lại mô hình:")
accuracy_optimal = accuracy_score(y_test, y_pred_optimal)
print(f"Accuracy on Test Set: {accuracy_optimal:.4f}")
print("Classification Report on Test Set:")
print(classification_report(y_test, y_pred_optimal))

cnf_matrix_optimal = metrics.confusion_matrix(y_test, y_pred_optimal)

class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

sns.heatmap(pd.DataFrame(cnf_matrix_optimal), annot=True, cmap="YlGnBu" ,fmt='g')
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Thực tế')
plt.xlabel('Dự báo')

```

Kết quả

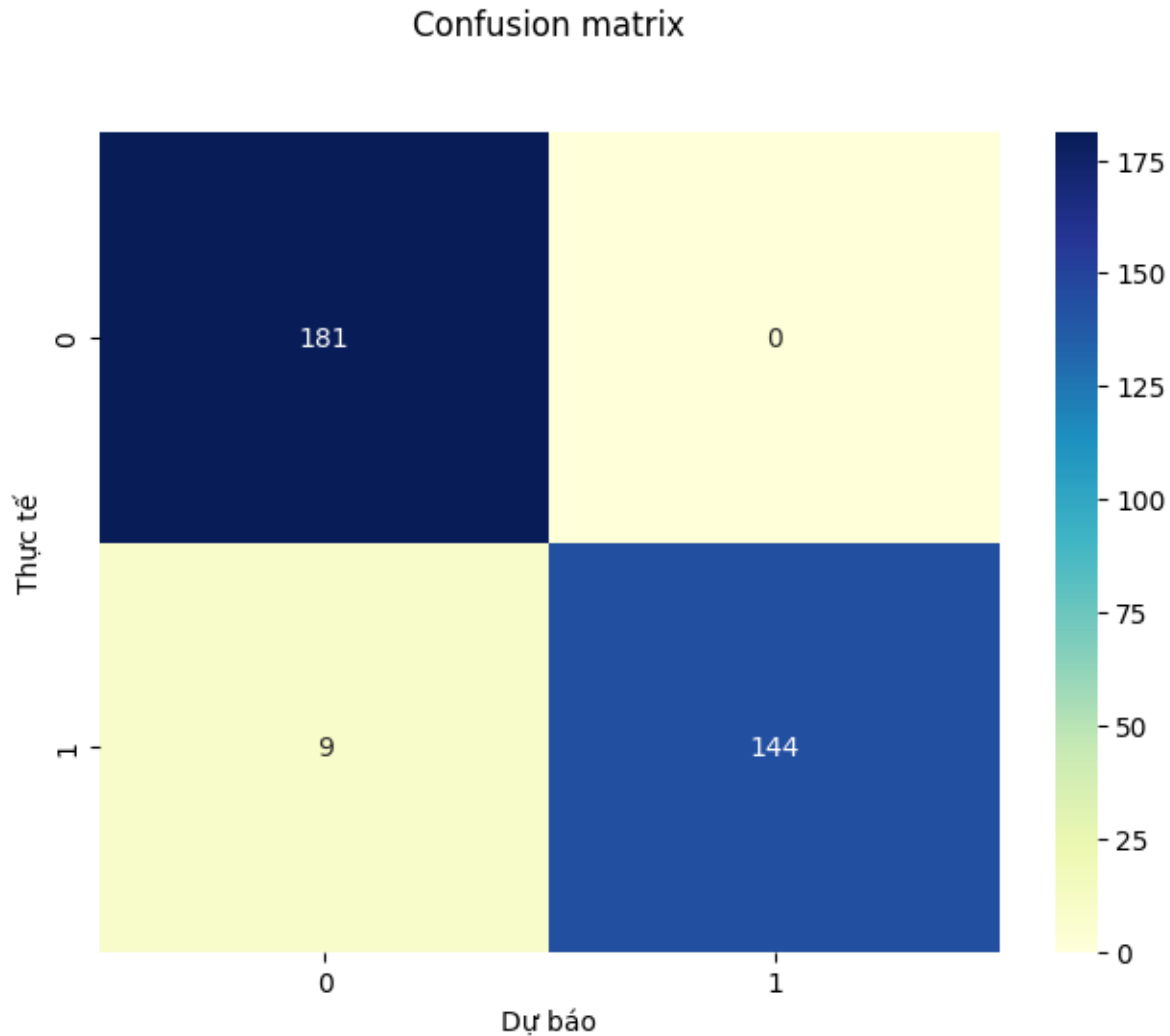
```

K tối ưu nhất cho mô hình : 13

Đánh giá lại mô hình:
Accuracy on Test Set: 0.9731
Classification Report on Test Set:

```

	precision	recall	f1-score	support
0	0.95	1.00	0.98	181
1	1.00	0.94	0.97	153
accuracy			0.97	334
macro avg	0.98	0.97	0.97	334
weighted avg	0.97	0.97	0.97	334



Dựa trên kết quả sau khi huấn luyện lại, có 1 vài nhận định như sau :

- Độ Accuracy lần này trên tập test tăng lên từ 96.71% lên 97.31%, cho thấy mô hình trên tập test có phần cải thiện tốt hơn, có thể dự đoán chính xác tốt hơn trên hầu hết tập dữ liệu.
- Độ Precision lần này của hai lớp 0 và 1 lần lượt là 95% và 100%, cho thấy các mẫu dữ liệu khi dự đoán các lớp 0 và 1, 95-100% sẽ đúng. Về lớp 1, các mẫu dữ liệu đều đoán đúng chính xác 100%.
- Độ Recall lần này của lớp 0 là 100%, của lớp 1 là 94%, cho thấy các mẫu dữ liệu thực tế là lớp 0, 100% được dự đoán hoàn toàn đúng, còn mẫu dữ liệu thực tế là lớp 1, 94% được dự đoán hoàn toàn đúng.
- Độ F1-Score của cả 2 lớp lần này tăng lên thành 97-98%, cho thấy mô hình hoạt động ổn định và tốt hơn sau khi huấn luyện lại mô hình trên cả 2 lớp, không có sự mất cân bằng trong việc dự đoán dữ liệu
- Về Macro Average và Weighted Average, cả 2 đều bằng 97%, cho thấy mô hình hoạt động đồng đều trên cả hai lớp và không có sự thiên vị về một lớp cụ thể

nào. Cộng với giá trị Support của lớp 0 và lớp 1 lần lượt là 181 và 153, cho thấy không xảy ra hiện tượng class imbalance.

So sánh giữa 2 mô hình KNN với 2 giá trị n-neighbors khác nhau, có thể kết luận rằng mô hình KNN với n-neighbors đã được tối ưu sẽ mạnh hơn so với mô hình với n-neighbors mặc định = 3. Lý do là vì Accuracy của n-neighbors tối ưu có phần tốt hơn, Precision đạt được 100% với lớp 1 và Recall đạt được 100% với lớp 0, F1-Score tốt hơn dẫn đến cân bằng hơn trong dự đoán dữ liệu.

Đánh giá chất lượng mô hình sử dụng cross validation với các độ đo precision, recall, f1 trên tập train, test. Kết luận mô hình được lựa chọn.

Để đánh giá chất lượng mô hình bằng cross-validation, nhóm đã làm như sau :

- Đầu tiên sẽ thiết lập các chỉ số đánh giá bằng hàm make_scorer và dictionary scoring
- Thực hiện cross-validation trên tập train với hàm cross_validate và chia thành 5-fold
- Đánh giá mô hình sau mỗi lần fold và tính toán các chỉ số cho mỗi lần fold
- Tính giá trị trung bình và độ lệch chuẩn của mỗi chỉ số qua 5 lần cross-validation
- Dùng 2 mô hình là Logistic Regression và K-Nearest Neighbors và dự đoán trên tập test
- Tính toán các chỉ số đánh giá bằng việc so sánh kết quả dự đoán với nhãn thực tế (y_test)

```
scoring = {
    'precision': make_scorer(precision_score),
    'recall': make_scorer(recall_score),
    'f1': make_scorer(f1_score),
    'accuracy': 'accuracy'
}

# Cross-validation trên tập huấn luyện
cv_results = cross_validate(regression, x_train_scaled, y_train, cv=5, scoring=scoring)

print("Cross-Validation Results on Training Data:")
for metric, values in scoring.items():
    mean_score = cv_results[f'test_{metric}'].mean()
    std_score = cv_results[f'test_{metric}'].std()
    print(f"{metric.capitalize()}: Mean = {mean_score:.4f}, Std = {std_score:.4f}")

# Đánh giá trên tập kiểm tra
y_test_pred = regression.predict(x_test_scaled)
precision = precision_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)
accuracy = accuracy_score(y_test, y_test_pred)

print("\nEvaluation on Test Data:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
```

```

scoring = {
    'precision': make_scorer(precision_score),
    'recall': make_scorer(recall_score),
    'f1': make_scorer(f1_score),
    'accuracy': 'accuracy'
}

# Cross-validation trên tập huấn luyện
cv_results = cross_validate(optimal_knn, x_train_scaled, y_train, cv=5, scoring=scoring)

print("Cross-Validation Results on Training Data:")
for metric, values in scoring.items():
    mean_score = cv_results[f'test_{metric}'].mean()
    std_score = cv_results[f'test_{metric}'].std()
    print(f"{metric.capitalize()}: Mean = {mean_score:.4f}, Std = {std_score:.4f}")

# Đánh giá trên tập kiểm tra
y_test_pred = optimal_knn.predict(x_test_scaled)
precision = precision_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)
accuracy = accuracy_score(y_test, y_test_pred)

print("\nEvaluation on Test Data:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

```

Kết quả

Logistic Regression :

```

Cross-Validation Results on Training Data:
Precision: Mean = 0.9799, Std = 0.0190
Recall: Mean = 0.9626, Std = 0.0112
F1: Mean = 0.9710, Std = 0.0077
Accuracy: Mean = 0.9700, Std = 0.0082

Evaluation on Test Data:
Accuracy: 0.9731
Precision: 0.9737
Recall: 0.9673
F1-Score: 0.9705

```

K-Nearest Neighbors :

Cross-Validation Results on Training Data:

Precision: Mean = 0.9911, Std = 0.0119

Recall: Mean = 0.9396, Std = 0.0188

F1: Mean = 0.9645, Std = 0.0087

Accuracy: Mean = 0.9640, Std = 0.0087

Evaluation on Test Data:

Accuracy: 0.9731

Precision: 1.0000

Recall: 0.9412

F1-Score: 0.9697

Dựa vào 2 kết quả này, có nhận định như sau :

- Trên tập train
 - Precision của KNN có phần tốt hơn so với Logistic ($0.99 > 0.98$)
 - Recall của Logistic có phần tốt hơn so với KNN ($0.96 > 0.94$)
 - F1-score của Logistic tốt hơn KNN ($0.97 > 0.96$)
 - Accuracy của Logistic tốt hơn KNN ($0.97 > 0.964$)
- Trên tập test
 - Accuracy của 2 mô hình đều ngang nhau ($=0.9731$)
 - Precision của KNN tốt hơn so với Logistic ($1 > 0.98$)
 - Recall của Logistic tốt hơn so với KNN ($0.97 > 0.94$)
 - F1-score của Logistic và KNN có phần ngang nhau
- Từ đó có thể thấy cả hai mô hình đều cho kết quả tốt, nhưng Logistic Regression có xu hướng cân bằng hơn giữa Precision và Recall hơn so với KNN, KNN bị mất cân bằng ở Recall (Precision đạt 100% trong khi Recall chỉ có 94%), F1-score của Logistic Regression ở cả 2 tập train và test đều tốt hơn KNN

Kết luận mô hình được lựa chọn là **LOGISTIC REGRESSION**.