

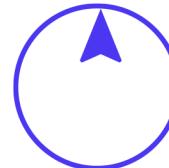
Marketing Campaign Classification

Nguyễn Hoàng Thái - Nhóm 4 - DA18

MINDX



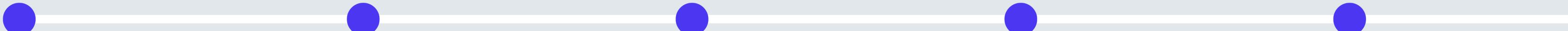
Tổng quan



Đây là bảng dữ liệu được thu thập được từ chiến dịch Marketing vào năm 2022 thể hiện về các khía cạnh của chiến dịch như: lượt phản hồi, thông tin khách hàng, thu nhập, hóa đơn,...

Mục tiêu

Tối đa hóa lợi nhuận ở chiến dịch Marketing kế tiếp.



01

Đọc và tìm hiểu dữ liệu

02

Làm sạch dữ liệu

03

Mã hóa và chuẩn hóa dữ liệu

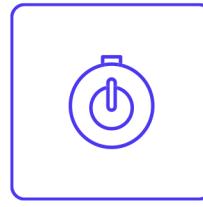
04

Phân nhóm khách hàng

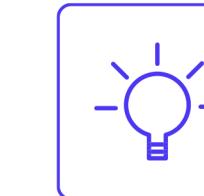
05

Dự báo cho chiến dịch Marketing kế tiếp

Mục lục



Phần 1:
Khai phá và làm sạch
dữ liệu



Phần 3:
Phân chia nhóm
khách hàng



Phần 2:
Mã hóa và chuẩn hóa
dữ liệu



Phần 4:
Dự báo sự phản hồi
chiến dịch tiếp theo



CÁC ĐẶC TRƯNG CỦA DỮ LIỆU

- AcceptedCmp1: khách hàng đồng ý với offer ở ngay chiến dịch quảng cáo đầu tiên.
- Response (target): khách hàng đồng ý với offer ở chiến dịch quảng cáo cuối cùng.
- DtCustomer: Ngày khách hàng tham gia với công ty.
- Education: trình độ học vấn của khách.
- Martial: tình trạng hôn nhân.
- Kidhome: số lượng con nhỏ
- Teenhome: số lượng teen của khách
- Income: thu nhập hàng năm
- NumDealsPurchases: số lần thanh toán có giảm giá
- NumCatalogPurchases: số lần thanh toán qua sử dụng ấn phẩm quảng cáo
- NumStorePurchases: số lần thanh toán trực tiếp tại cửa hàng.
- NumWebPurchases: số lần thanh toán qua website
- Recency: số ngày tính từ lần gần nhất thanh toán.



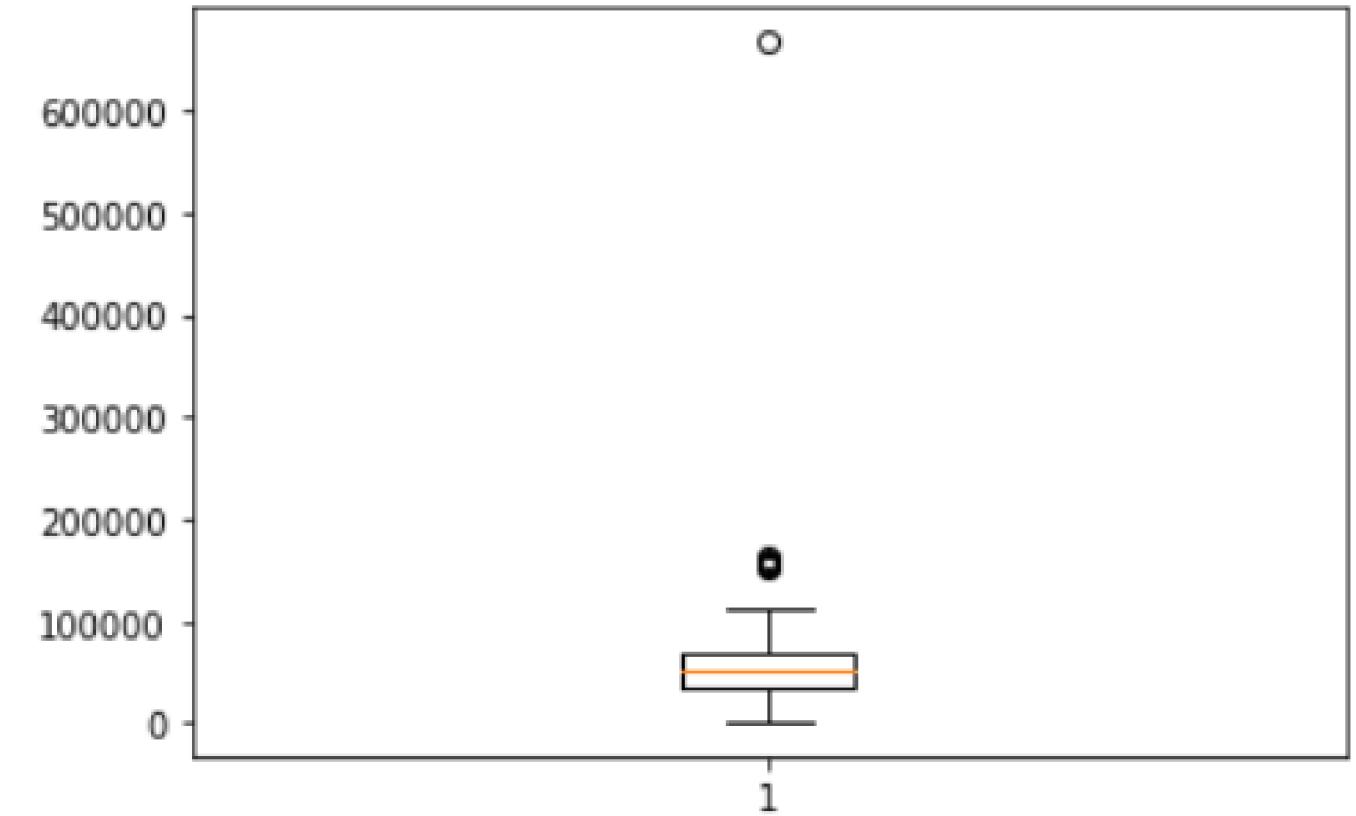
Phần 1:



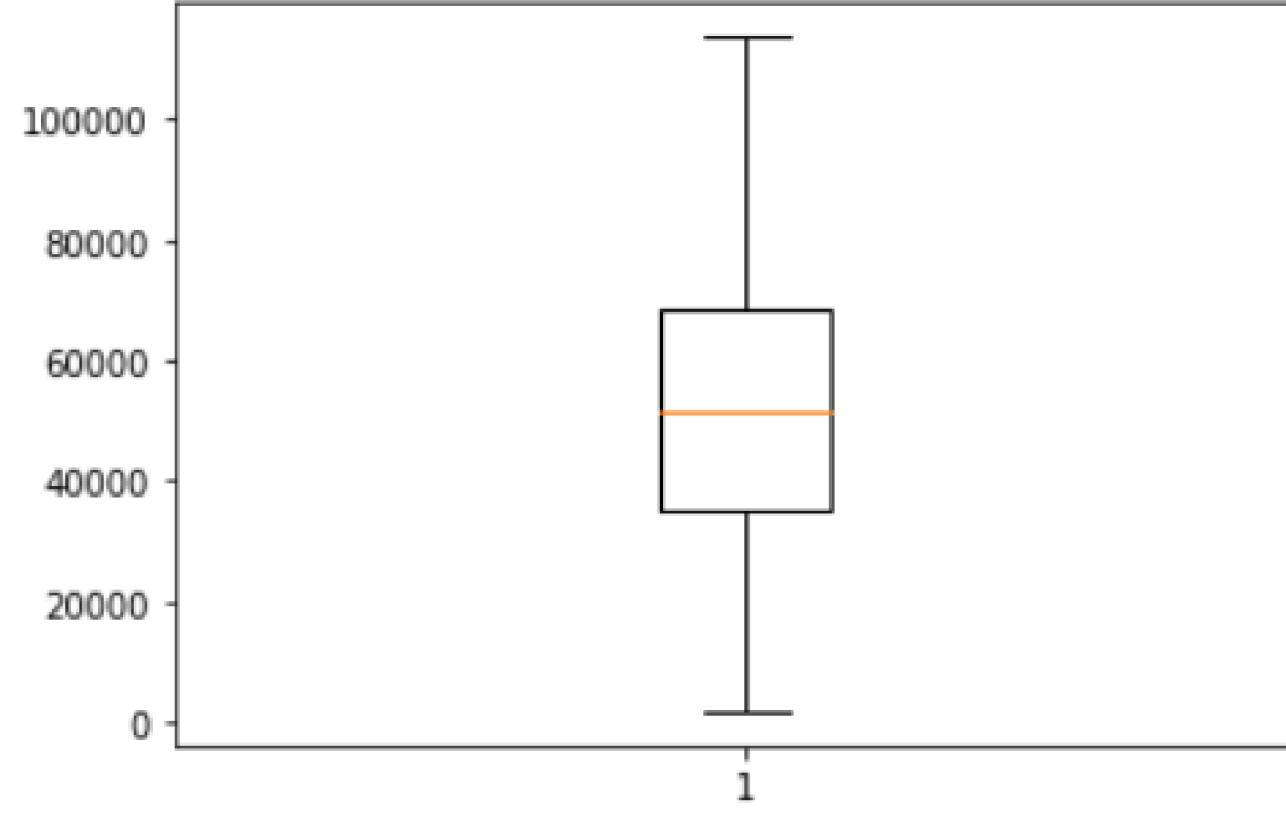
Khai phá và làm sạch
dữ liệu

Loại bỏ outliers

```
plt.boxplot(df['Income'])  
plt.show()
```



```
plt.boxplot(df['Income'])  
plt.show()
```



```
Q1 = df['Income'].quantile(0.25)  
Q3 = df['Income'].quantile(0.75)  
IQR = Q3 - Q1  
outliers = df[(df['Income'] < (Q1 - 1.5 * IQR)) | (df['Income'] > (Q3 + 1.5 * IQR))]  
print("Number of outliers in the Income column:", len(outliers))
```

Number of outliers in the Income column: 8

```
df = df[~((df['Income'] < (Q1 - 1.5 * IQR)) | (df['Income'] > (Q3 + 1.5 * IQR)))]  
print('Update shape of the dataframe:', df.shape)
```

Update shape of the dataframe: (2208, 29)

Dựa vào biểu đồ hộp về dữ liệu thu nhập của khách hàng trong cột 'Income' thấy có rất nhiều Outliers
=> Sử dụng IQR để lọc Outliers

Tạo thêm cột

```
print('Unique values in Education column:', df['Education'].unique())
print('Unique values in Marital_Status column:', df['Marital_Status'].unique())

Unique values in Education column: ['Graduation' 'PhD' 'Master' 'Basic' '2n Cycle']
Unique values in Marital_Status column: ['Single' 'Together' 'Married' 'Divorced' 'Widow' 'Alone' 'Absurd' 'YOLO']

def education_level(education):
    if education in ['Graduation', 'PhD', 'Master']:
        return 'High'
    if education in ['Basic']:
        return 'Middle'
    else:
        return 'Low'

df['Education_Level'] = df['Education'].apply(education_level)

def living_status(marital_status):
    if marital_status in ['Alone', 'Absurd', 'YOLO']:
        return 'Living Alone'
    else:
        return 'Living with Others'

df['Living_Status'] = df['Marital_Status'].apply(living_status)
```

- Tạo thêm cột 'Education_Level' với phân loại mới: 'High', 'Middle', 'Low'.
- Tạo thêm cột 'Living_Status' với phân loại mới: 'Living Alone' và 'Living with others'

```
df['Age'] = 2023 - df['Year_Birth']
df['Total_Campaign_Accepted'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1)
df['Average_Spend'] = (df[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']].sum(axis=1)) / df['NumDealsPurchases']
df['Spent'] = df['MntWines']+df["MntFruits"]+df[ 'MntMeatProducts'] +df[ 'MntFishProducts']+df[ 'MntSweetProducts']+ df[ 'MntGoldProds']
df['Is_Parent'] = (df['Kidhome'] + df['Teenhome'] > 0).astype(int)
df['total_spending'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts'] + df['MntSweetProducts'] + df['MntGoldProds']
df['avg_web_visits'] = df['NumWebVisitsMonth'] / 12
df['online_purchase_ratio'] = df['NumWebPurchases'] / (df['NumWebPurchases'] + df['NumCatalogPurchases'] + df['NumStorePurchases'])
```

- Tạo thêm các cột tính toán mới cho phân tích sau.

Tạo thêm cột

```
print('Unique values in Education column:', df['Education'].unique())
print('Unique values in Marital_Status column:', df['Marital_Status'].unique())

Unique values in Education column: ['Graduation' 'PhD' 'Master' 'Basic' '2n Cycle']
Unique values in Marital_Status column: ['single' 'Together' 'Married' 'Divorced' 'Widow' 'Alone' 'Absurd' 'YOLO']

def education_level(education):
    if education in ['Graduation', 'PhD', 'Master']:
        return 'High'
    if education in ['Basic']:
        return 'Middle'
    else:
        return 'Low'

df['Education_Level'] = df['Education'].apply(education_level)

def living_status(marital_status):
    if marital_status in ['Alone', 'Absurd', 'YOLO']:
        return 'Living Alone'
    else:
        return 'Living with Others'

df['Living_Status'] = df['Marital_Status'].apply(living_status)
```

- Tạo thêm cột 'Education_Level' với phân loại mới: 'High', 'Middle', 'Low'.
- Tạo thêm cột 'Living_Status' với phân loại mới: 'Living Alone' và 'Living with others'

```
df['Age'] = 2023 - df['Year_Birth']
df['Total_Campaign_Accepted'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1)
df['Average_Spend'] = (df[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']].sum(axis=1)) / df['NumDealsPurchases']
df['Spent'] = df['MntWines']+df['MntFruits']+ df['MntMeatProducts'] +df['MntFishProducts']+df['MntSweetProducts']+ df['MntGoldProds']
df['Is_Parent'] = (df['Kidhome'] + df['Teenhome'] > 0).astype(int)
df['total_spending'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts'] + df['MntSweetProducts'] + df['MntGoldProds']
df['avg_web_visits'] = df['NumWebVisitsMonth'] / 12
df['online_purchase_ratio'] = df['NumWebPurchases'] / (df['NumWebPurchases'] + df['NumCatalogPurchases'] + df['NumStorePurchases'])
```

- Tạo thêm các cột tính toán mới cho phân tích sau.

```
df = df[df['Average_Spend'] != float('inf')]
df = df[df['avg_web_visits'] != float('inf')]
df = df[df['online_purchase_ratio'] != float('inf')]
df['Income'] = df['Income'].astype('int64')
```

- Xóa các giá trị inf ở các cột tính toán

```
df['Average_Spend'] = round(df['Average_Spend'], 2)
df['avg_web_visits'] = round(df['avg_web_visits'], 2)
df['online_purchase_ratio'] = round(df['online_purchase_ratio'], 2)
```

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

- Làm tròn số ở các cột tính toán
- Xóa các giá trị NaN

Mã hóa và chuẩn hóa

**Mã
hóa**

Chuyển các dữ liệu dạng
'object' sang kiểu dữ liệu
dạng số

**Chuẩn
hóa**

Nâng cao hiệu suất của các
Model

Phần 2:



Mã hóa và chuẩn hóa
dữ liệu

Mã hóa

```
df['Education_Level'].replace(to_replace=['Low','Middle','High'], value=[0,1,2],inplace=True)
df['Living_Status'].replace(to_replace=['Living Alone','Living with Others'], value=[0,1],inplace=True)
```

- Mã hóa dữ liệu hai cột 'Education_Level' và 'Living_Status' về dữ liệu dạng số.

Chuẩn hóa

```
from sklearn.preprocessing import StandardScaler #Thư viện
X_std = StandardScaler() #Gọi thư viện ra đặt tên biến cho nó
X_std = X_std.fit_transform(df)
#X_std
```

- Sử dụng thư viện 'StandardScaler' để chuẩn hóa do dữ liệu đã được loại bỏ outliers.

SỬ DỤNG KMEANS CLUSTERING ĐỂ PHÂN NHÓM KHÁCH HÀNG

Tìm k bằng giá trị SSE



Tìm k bằng silhouette coefficents



Tìm k với giá trị wcss



Phần 3:
Phân nhóm khách hàng



Phần 3: Phân nhóm khách hàng

```
from kneed import KneeLocator
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans

kmeans_kwargs = {}
sse = [] #Tính toán giá trị SSE cho mỗi giá trị của k
for k in range(2, 20):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(scaled_features)
    sse.append(kmeans.inertia_)
```

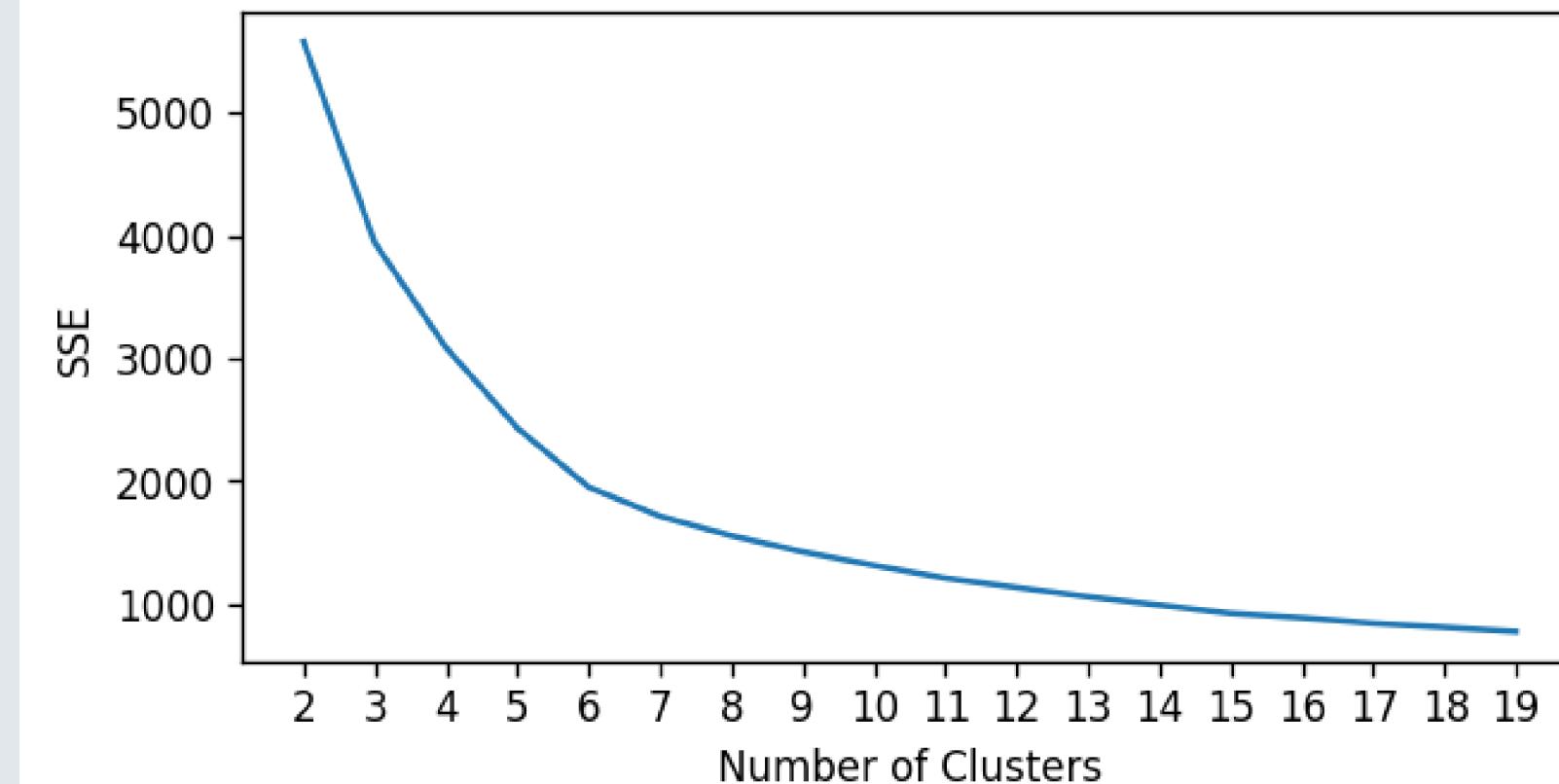
- Import thư viện 'KneeLocator' để tìm đoạn gấp.
- Tính toán giá trị k theo SSE từ 2 -> 20

```
| k1 = KneeLocator(range(2, 20), sse, curve="convex", direction="decreasing") #Tính toán, hiển thị ra giá trị elbow
| k1.elbow
```

6

- Giá trị k theo phương pháp SSE là 6

Biểu đồ giá trị SSE

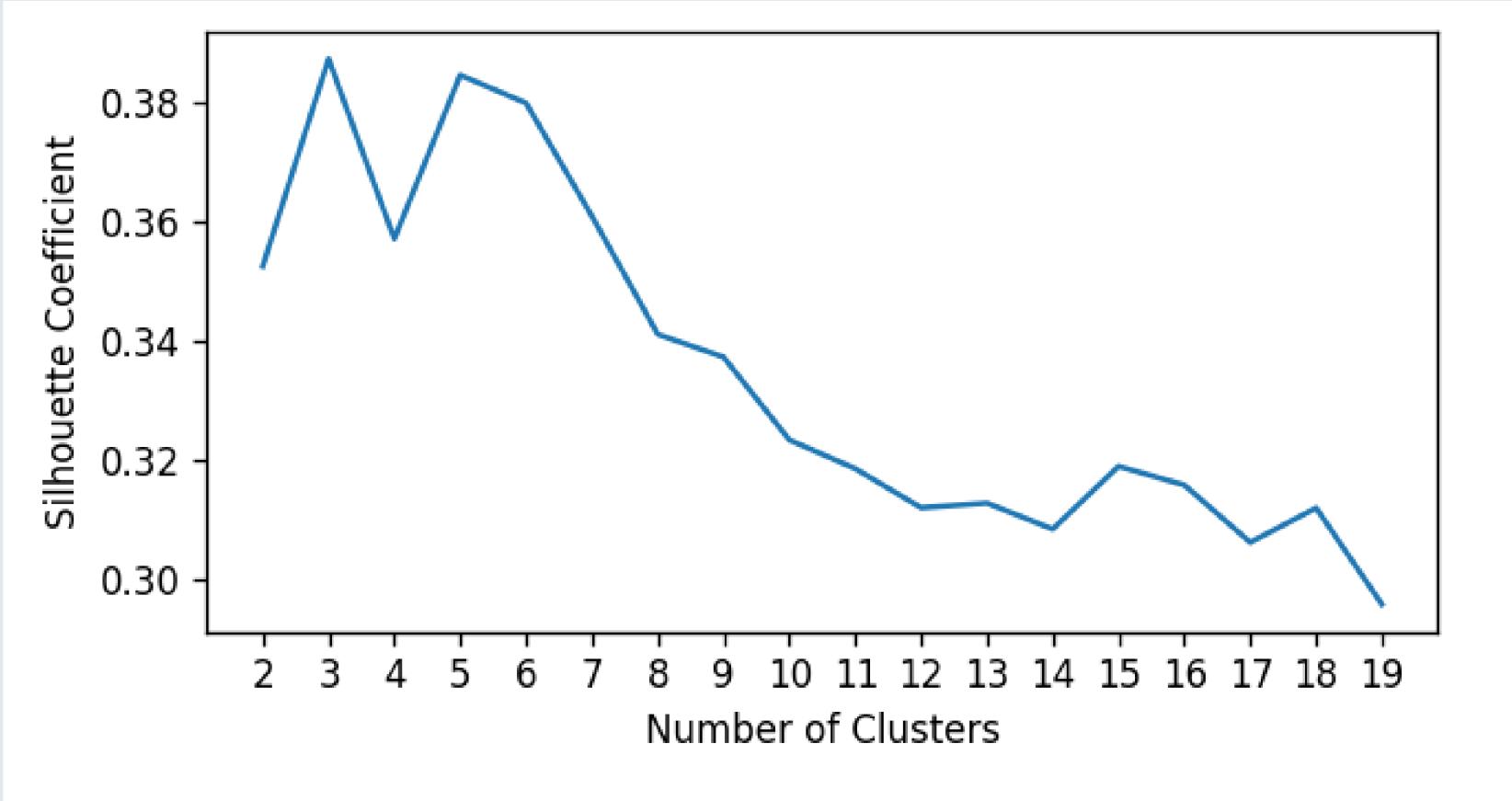




Phần 3: Phân nhóm khách hàng

```
#silhouette
silhouette_coefficients = []
for k in range(2, 20):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(scaled_features)
    score = silhouette_score(scaled_features, kmeans.labels_)
    silhouette_coefficients.append(score)
```

Biểu đồ giá trị silhouette



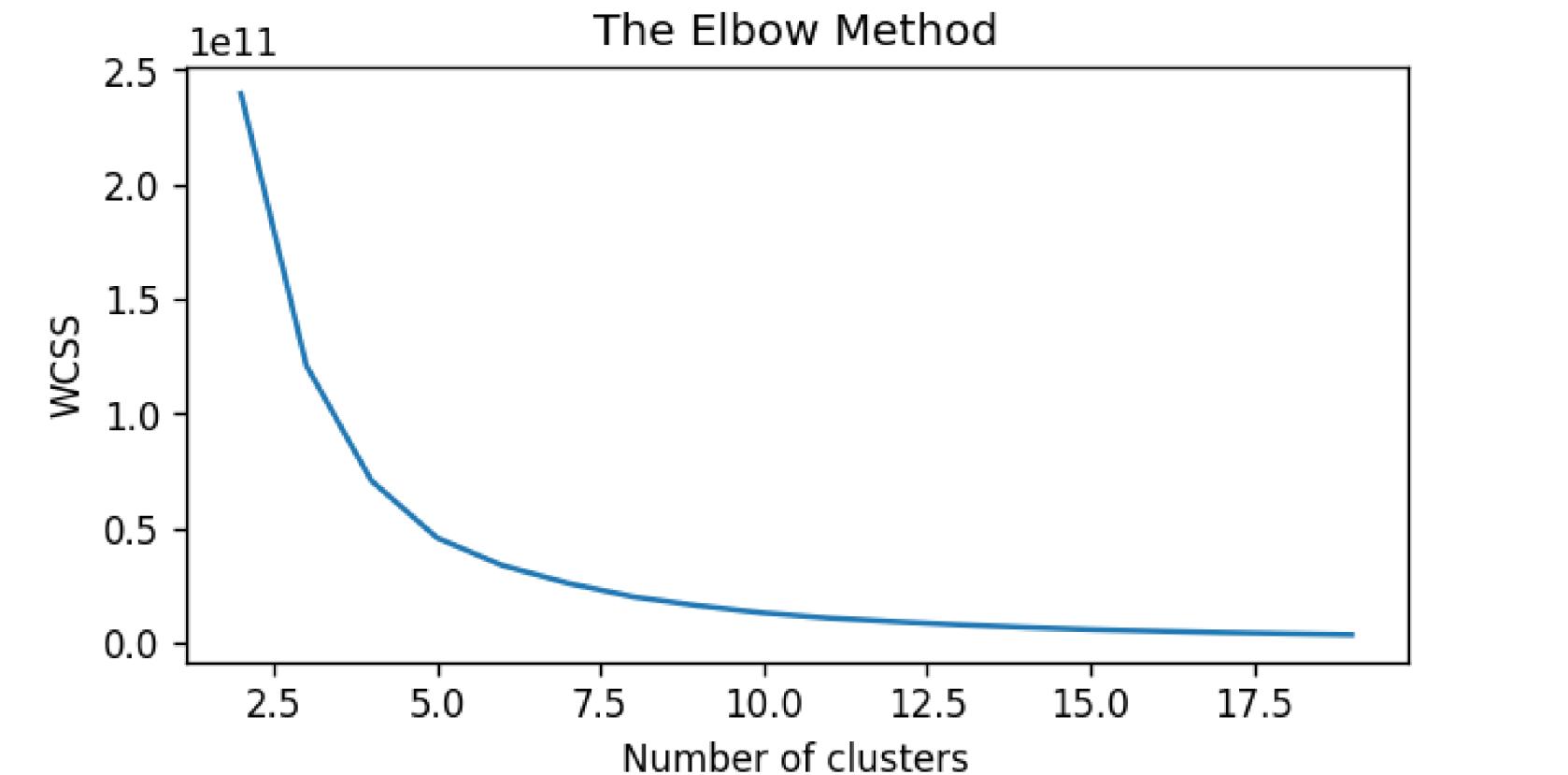
- Sử dụng thư viện 'silhouette_score' tính.
- Tính toán giá trị k theo silhouette từ 2 -> 20
- Giá trị k theo phương pháp silhouette là 3



Phần 3: Phân nhóm khách hàng

```
#WCSS
wcss = []
for i in range(2, 20):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X1)
    wcss.append(kmeans.inertia_)
plt.plot(range(2, 20), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Biểu đồ giá trị WCSS



Tính toán giá trị k theo WCSS từ 2 -> 20

```
[ ] kmeans = KMeans(n_clusters = 4)
y_pred = kmeans.fit_predict(X1)

[ ] df['Class']=y_pred

[ ] df['Class'].replace(to_replace=[0,1,2,3], value=['Member', 'Gold', 'Platinum', 'Diamond'], inplace=True)
```

- Giá trị k theo phương pháp WCSS là k = 3 -> 5
=> Chọn giá trị k =4 cho model K means



Phần 3: Phân nhóm khách hàng

Bảng data phân nhóm khách hàng mẫu sau phân nhóm

		ID	Income	Kidhome	Teenhome	Recency	Complain	Response	Education_Level	Living_Status	Age	Total_Campaign_Accepted	Average_Spend	Spent	Is_Parent	total_spending	avg_web_visits	online_purchase_ratio	Class
2220	6261	58025	0	1	81	0	0	0	2	1	44	0	156.67	740	1	470	0.33	0.20	Member
1573	8135	27161	1	0	41	0	0	0	2	1	37	0	52.00	59	1	52	0.58	0.25	Platinum
2049	2079	81044	0	0	0	0	0	0	0	1	76	0	1208.00	1658	0	1208	0.08	0.24	Diamond
416	6504	19789	1	0	24	0	0	0	0	1	48	1	26.00	80	1	78	0.42	0.33	Platinum
100	1473	47823	0	1	0	0	0	0	0	1	63	0	36.00	125	1	72	0.67	0.40	Gold
815	11176	65968	0	1	12	0	0	0	2	1	53	0	247.50	871	1	495	0.25	0.31	Member
993	7646	64449	1	0	70	0	0	0	2	1	34	0	201.25	1023	1	805	0.33	0.25	Member
1333	5147	90842	0	0	57	0	0	0	2	1	75	0	1424.00	2198	0	1424	0.08	0.15	Diamond
1356	7912	38136	1	0	69	0	0	0	2	1	45	1	51.50	111	1	103	0.67	0.50	Gold
2169	4548	41967	1	1	66	0	0	0	2	1	42	0	54.00	77	1	54	0.33	0.25	Gold

Khách hàng được chia thành 4 class là: 'Member', 'Gold', 'Platinum', 'Diamond'.

**Góp phần cải thiện hiệu suất của
chiến dịch Marketing kế tiếp bằng
cách cung cấp giải pháp sự phản
hồi của khách hàng.**



Phần 4:



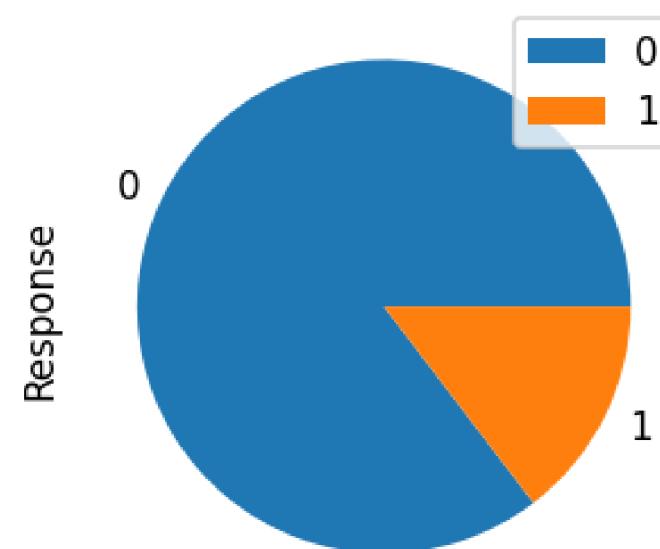
Dự báo sự phản hồi
chiến dịch tiếp theo



Phần 4: Dự báo

```
df1 = df.drop(['Class'],axis=1)

df1['Response'].value_counts().plot.pie(legend = ["0", "1"])
```



```
#Chia lại X, y và chạy mô hình
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report
X_train, X_test, y_train, y_test = train_test_split(X_SM, y_SM, test_size=0.3, random_state=0)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred= model.predict(X_test)
print(classification_report(y_test, y_pred))
```

- Dữ liệu cột 'Response' cho thấy data bị mất cân bằng
- Sử dụng phương pháp SMOTE (Synthetic Minority Over Sampling) để cân bằng lại data
- Chia X,y để chạy model LogisticRegression cho tỉ lệ chính xác là 65%

```
#Oversampling
# SMOTE (Synthetic Minority Over-sampling)

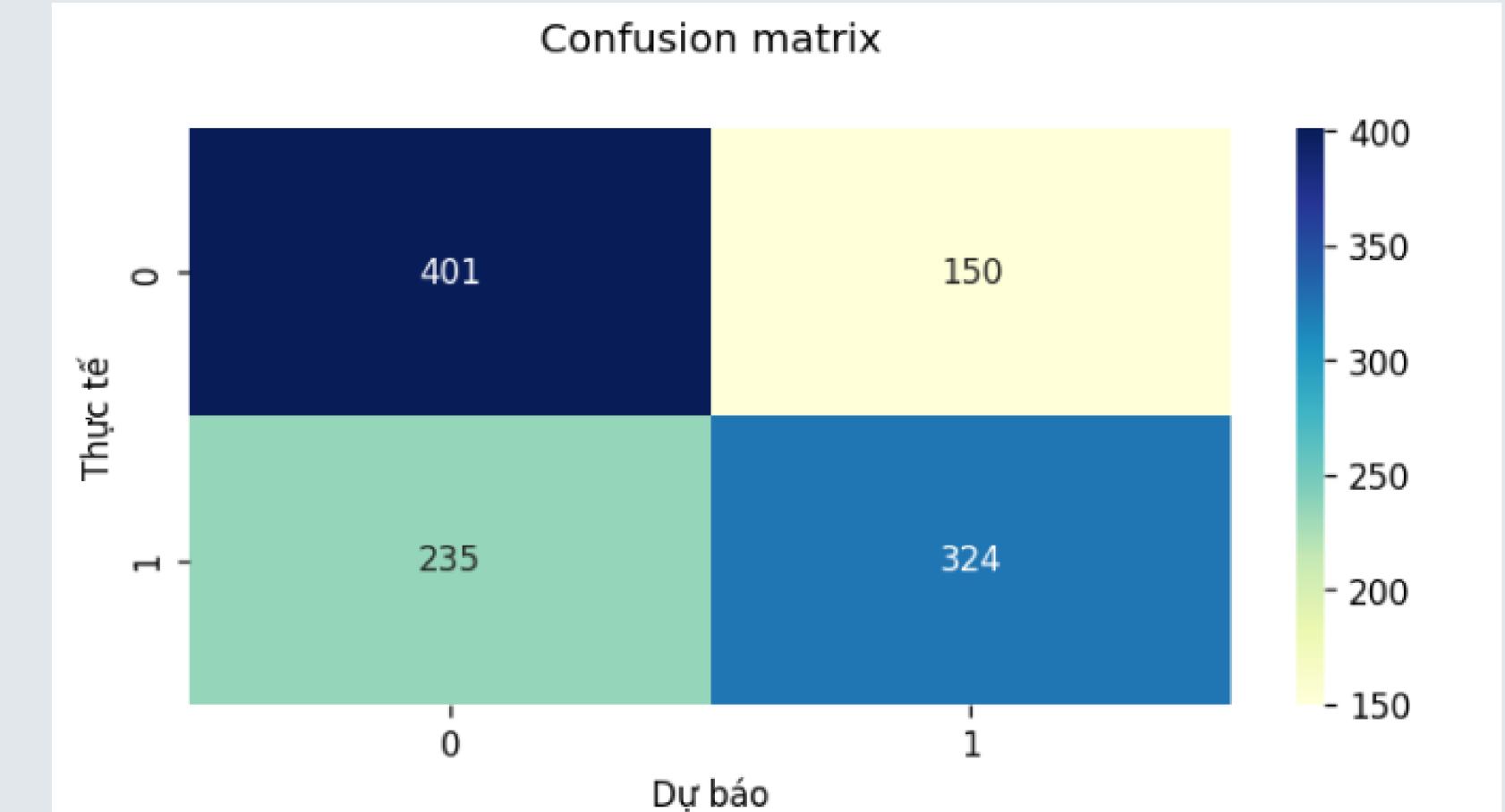
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import RandomOverSampler
y = df1['Response'].values #Load lại X, y từ bảng data gốc
X_cols = ['Income', 'Kidhome', 'Teenhome', 'Recency', 'Complain', 'Education_Level', 'Living_Status', 'Age', 'Total_Campaign_Accepted', 'Average_Spend', 'Spent', 'Is_Parent', 'total_spending', 'avg_web_visits', 'online_purchase_ratio']
X = df1.loc[:,X_cols].values
SM_Model=SMOTE()
ROS_Model=RandomOverSampler()
X_SM, y_SM=ROS_Model.fit_resample(X,y)
X_SM.shape, y_SM.shape

((3700, 15), (3700,))
```



Phần 4: Dự báo

```
import seaborn as sns
import numpy as np
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
class_names=[0,1] # name of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
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')
```



- Import thư viện 'seaborn', 'numpy'.
- Tạo confusion matrix để hình dung hiệu suất của mô hình.



Phần 4: Dự báo

```
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.tree import DecisionTreeClassifier  
ada = AdaBoostClassifier(LogisticRegression())  
ada.fit(X_train, y_train)  
accuracy_ada = ada.score(X_test, y_test)  
  
print("Accuracy: {:.2f}".format(accuracy_ada))
```

AdaBoost cho tỉ lệ chính xác là 68%

```
from sklearn.ensemble import GradientBoostingClassifier  
Gra = GradientBoostingClassifier()  
Gra.fit(X_train, y_train)  
accuracy_gra = Gra.score(X_test, y_test)  
print("Accuracy: {:.2f}".format(accuracy_gra))
```

Accuracy: 0.85

GradientBoosting cho tỉ lệ chính xác là 85%

```
import xgboost as xgb  
Xg = xgb.XGBClassifier()  
Xg.fit(X_train, y_train)  
accuracy_xg = Xg.score(X_test, y_test)  
print("Accuracy: {:.2f}".format(accuracy_xg))
```

Accuracy: 0.95

XGBoost cho tỉ lệ chính xác là 95%

```
import lightgbm as lgb  
Lig = lgb.LGBMClassifier()  
Lig.fit(X_train, y_train)  
accuracy_lig = Lig.score(X_test, y_test)  
print("Accuracy: {:.2f}".format(accuracy_lig))
```

Accuracy: 0.95

LightGBM cho tỉ lệ chính xác là 95%

Cảm ơn bạn!