DETERMINING IDEAL CUSTOMER PROFILE

Reg # 2024-MS-DS-118 and 2024-MS-DS-119

Determining Ideal Customer Profile for a departmental store

Contents

1. Introduction

Problem Statement

The departmental store aims to enhance its customer engagement and marketing effectiveness. However, the store faces two key challenges:

- Identifying the Ideal Customer Profile: The store needs to understand the characteristics of its most valuable customers to target them more effectively. Without a clear profile, marketing efforts may be misaligned, leading to lower conversion rates and lower ROI and inefficient resource allocation.

To address this challenge, the store aims to develop an ideal customer profile using a data-driven approach in the first phase. The second phase will focus on building a predictive model to forecast which campaign a customer is most likely to accept an offer.

This project pertains to the first phase, and its end result will be:

- Data-driven segmentation that distinguishes ideal customers from the broader customer base, enabling more focused customer engagement.
- A well-defined ideal customer profile that identifies the key characteristics of high-value customers.

Success Criteria

The success of this project will be measured using the following criteria:

- Exploration and preparation of data
- Customer segmentation & analysis
- Identification of ideal customer profile using tree induction model

Data Science Team:

- Syed Muhammad Baqar Shah, Reg # 2024-MS-DS-119

2. Data Collection

The data for this project is sourced from Kaggle, a well-known platform for open datasets. The dataset contains information related to customer demographics, purchase history, and marketing campaign interactions. This data will serve as the foundation for building models to identify the ideal customer profile.

Data Source: [https://www.kaggle.com/datasets/jackdaoud/marketing-data]

Key Components of the Dataset:

Sr #	Feature Title	Feature Description
1	AcceptedCmp1	1 if customer accepted the offer in the 1st campaign, 0 otherwise
2	AcceptedCmp2	1 if customer accepted the offer in the 2nd campaign, 0 otherwise

3	AcceptedCmp3	1 if customer accepted the offer in the 3rd campaign, 0 otherwise
4	AcceptedCmp4	1 if customer accepted the offer in the 4th campaign, 0 otherwise
5	AcceptedCmp5	1 if customer accepted the offer in the 5th campaign, 0 otherwise
6	Response (target)	1 if customer accepted the offer in the last campaign, 0 otherwise
7	Complain	1 if customer complained in the last 2 years
8	DtCustomer	Date of customer's enrolment with the company
9	Education	Customer's level of education
10	Marital	Customer's marital status
11	Kidhome	Number of small children in customer's household
12	Teenhome	Number of teenagers in customer's household
13	Income	Customer's yearly household income
14	MntFishProducts	Amount spent on fish products in the last 2 years
15	MntMeatProducts	Amount spent on meat products in the last 2 years
16	MntFruits	Amount spent on fruits products in the last 2 years
17	MntSweetProducts	Amount spent on sweet products in the last 2 years
18	MntWines	Amount spent on wine products in the last 2 years
18	MntGoldProds	Amount spent on gold products in the last 2 years
20	NumDealsPurchases	Number of purchases made with discount
21	NumCatalogPurchases	Number of purchases made using catalogue
22	NumStorePurchases	Number of purchases made directly in stores
23	NumWebPurchases	Number of purchases made through company's web site
24	NumWebVisitsMonth	Number of visits to company's web site in the last month
25	Recency	Number of days since the last purchase
26	Z_CostContact	Cost to reach a customer in each campaign
27	Z_Revenue	Revenue a customer brings if converted

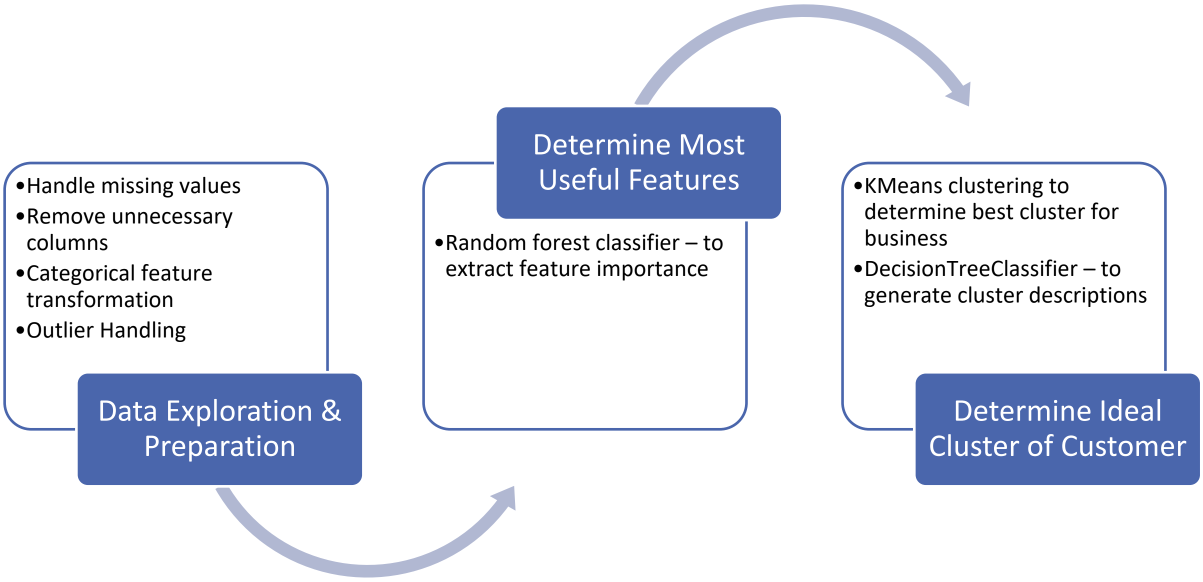
Notes:

The features ***Z_CostContact*** and ***Z_Revenue*** are not explicitly defined in the source dataset's feature descriptions. The provided definitions are based on logical assumptions derived from the context of the other features.

3. Project Methodology

The project is divided into two distinct parts according to the project requirements. The pipelines for both parts are illustrated in the diagrams below.

Pipeline for determining Ideal Customer Profile:



4. Data Exploration and Preparation

Original Data Information

This is how the original data looks like:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  AcceptedCmp3         2240 non-null   int64  
 21  AcceptedCmp4         2240 non-null   int64  
 22  AcceptedCmp5         2240 non-null   int64  
 23  AcceptedCmp1         2240 non-null   int64  
 24  AcceptedCmp2         2240 non-null   int64  
 25  Complain             2240 non-null   int64  
 26  Z_CostContact        2240 non-null   int64  
 27  Z_Revenue            2240 non-null   int64  
 28  Response             2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

The initial analysis of the data reveals a mix of categorical and numerical features. The categorical features include ***Education*** and ***Marital_Status***.

The ***Year_Birth*** feature will be transformed into ***Age***, while the ***Dt_Customer*** feature, currently stored as an object, will be converted into the number of days since the customer joined.

Handling Missing Values

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

```
ID                      0
Year_Birth              0
Education               0
Marital_Status          0
Income                 24
Kidhome                 0
Teenhome                0
Dt_Customer             0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         0
MntFishProducts         0
MntSweetProducts        0
MntGoldProds            0
NumDealsPurchases       0
NumWebPurchases         0
NumCatalogPurchases     0
NumStorePurchases       0
NumWebVisitsMonth       0
AcceptedCmp3            0
AcceptedCmp4            0
AcceptedCmp5            0
AcceptedCmp1            0
AcceptedCmp2            0
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
```

The dataset contains no null values except for the ***Income*** feature, which has 24 missing entries. Since removing these entries will not significantly impact the analysis, these will be dropped instead of being filled.

```
df.dropna(subset=['Income'], inplace=True)
df.isna().sum()
```

```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
```

Duplicate Records Handling

```
df[df['ID'].duplicated()]
```

ID	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	...	Marital_Status_Divorced	Marital_

0 rows × 40 columns

All customer records are unique, with no duplicate entries in the dataset.

Dropping Irrelevant Features

The features ***Z_CostContact*** and ***Z_Revenue*** are not relevant for both the objectives of the project and hence would be dropped.

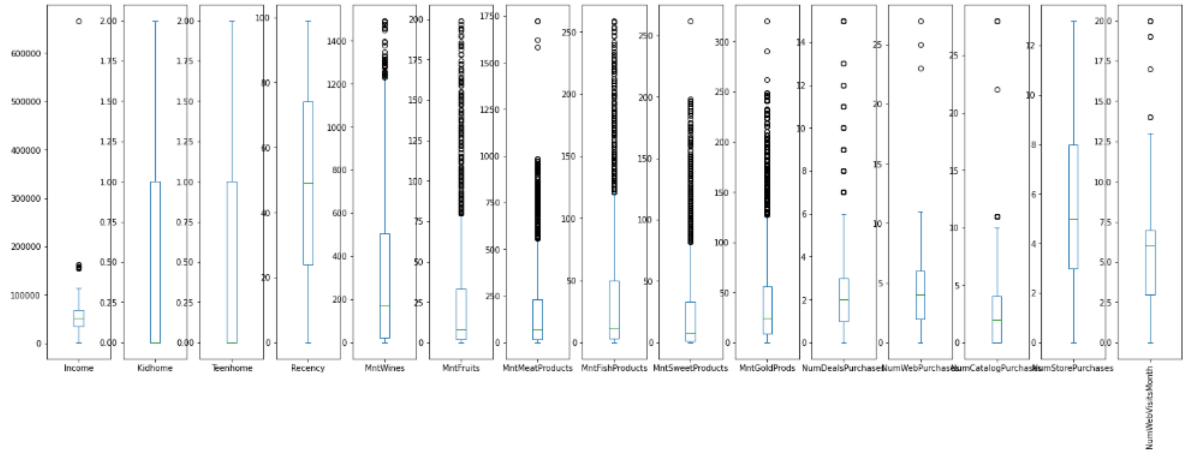
```
df.drop(['Z_CostContact', 'Z_Revenue'], axis=1, inplace=True)
```

Outliers Handling

After addressing the missing records, checking for duplicates, and removing irrelevant features, I reviewed the data to identify any potential outliers. For this analysis, I focused on a subset of relevant features, excluding the Customer ***ID*** and Boolean features (such as ***AcceptedCmp{X}***), as they are not relevant for outlier detection.

```
select_cols = ['Income', 'Kidhome', 'Teenhome', 'Recency', 'MntWines',
        'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
        'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
        'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth']
```

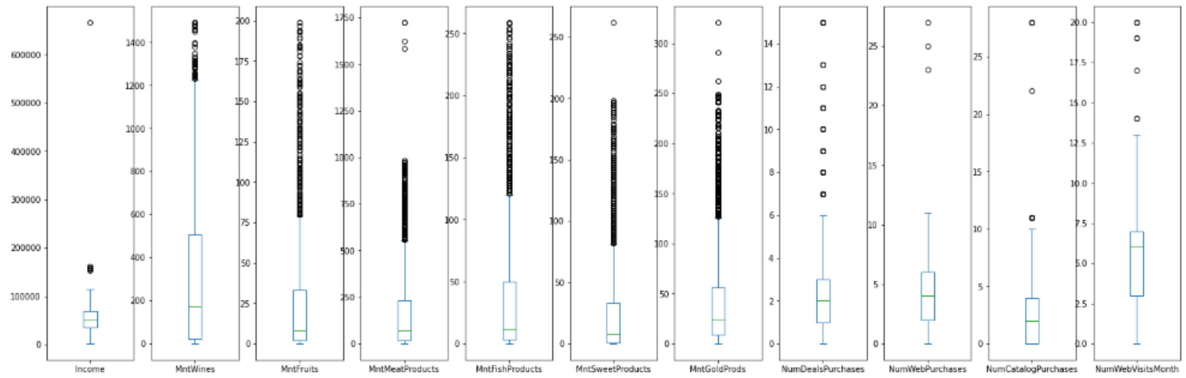
```
df[select_cols].plot(kind='box', subplots=True, sharex=False, sharey=False, figsize=(25, 8))
plt.xticks(rotation=90)
plt.show()
```



I observed that there are no outliers in the ***Kidhome***, ***Teenhome***, ***Recency***, ***NumStorePurchases***, ***Total_Spent***, and ***Days_Since_Customer*** features. Therefore, I will remove these features from the chart to focus on those that exhibit outliers.

```
select_cols = ['Income', 'MntWines','MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
        'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumWebVisitsMonth']
```

```
df[select_cols].plot(kind='box', subplots=True, sharex=False, sharey=False, figsize=(25, 8))
plt.show()
```



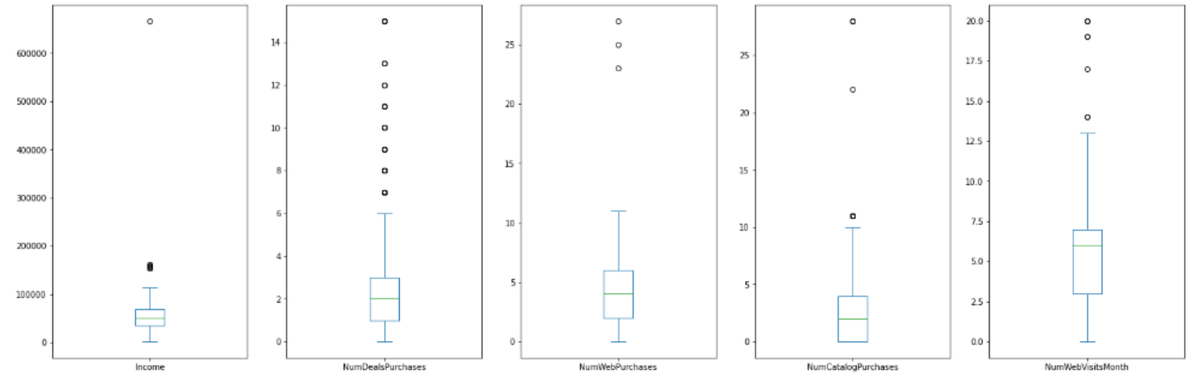
I understand that removing outliers from the data alters its distribution. When plotting a box plot on the new distribution, it may reveal additional outliers. Continuously eliminating outliers would lead to significant data loss. Therefore, I will avoid removing outliers from features where their presence makes intuitive sense.

With that said, I will remove the outliers from the following features: ***Income***, ***NumDealsPurchases***, ***NumWebPurchases***, ***NumCatalogPurchases***, ***NumWebVisitsMonth***.

```
cols = ['Income', 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumWebVisitsMonth']
for col in cols:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3-q1
    upper_bound = q3+(2*iqr)
    lower_bound = q1 - (2*iqr)
    no_outlier_df = df[(df[col]<=upper_bound) & df[col] >=lower_bound]
```

Box plot after outlier removal:

```
no_outlier_df[cols].plot(kind='box', subplots=True, sharex=False, sharey=False, figsize=(25, 8))
plt.show()
```



As mentioned earlier, removing outliers from the data alters its distribution, and plotting the box plot confirms that new outliers have emerged from the updated distribution. From the plot, it is evident that the ***Income*** column contains a few extreme outliers (salaries above $600,000) that significantly skew the data. Let's now check how many instances have a salary above $600,000.

```
no_outlier_df[no_outlier_df['Income']>600000].shape[0]
```

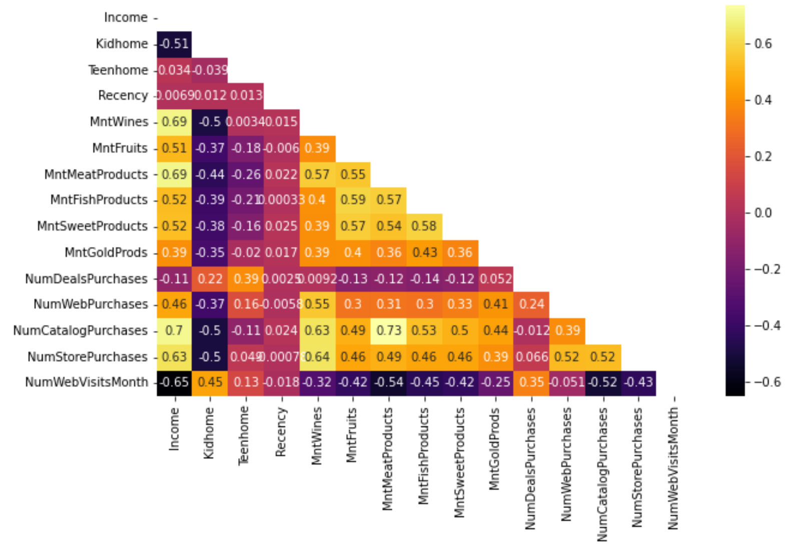
```
1
```

There is only one such instance and removing it from the data makes sense.

```
no_outlier_df = no_outlier_df[no_outlier_df['Income']<600000]
```

Correlation between Features

```
corr_matrix = no_outlier_df[selected_cols].select_dtypes(include=np.number).corr()
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, mask=mask, annot=True, cmap='inferno')
plt.show()
```



There is no strong correlation between the features themselves.

Data Transformation

Feature Engineering

```
# transform Year_Birth to age
no_outlier_df['age'] = no_outlier_df['Year_Birth'].apply(lambda x: 2024-x)
no_outlier_df.drop(['Year_Birth'], axis=1, inplace=True)

# transform Kidhome and Teenhome to one columns: n_kids
no_outlier_df['n_kids'] = no_outlier_df['Kidhome']+no_outlier_df['Teenhome']

# transform the Dt_customer to number of days from the cutoff date of 12-Dec-24
no_outlier_df['Dt_Customer'] = pd.to_datetime(no_outlier_df['Dt_Customer'])
no_outlier_df['days_since_customer'] = no_outlier_df['Dt_Customer'].apply(lambda x: datetime(2024, 12, 12)-x)
no_outlier_df['days_since_customer'] = no_outlier_df['days_since_customer'].dt.days
no_outlier_df.drop(['Dt_Customer'], axis=1, inplace=True)

# calculate the total amount spent by the customer throughout the lifecycle for all the categories. We will create a new column f
no_outlier_df['total_spent'] = no_outlier_df['MntWines'] + no_outlier_df['MntFruits'] + no_outlier_df['MntMeatProducts'] + no_out
```

Encoding Categorical Features

```
no_outlier_df = pd.get_dummies(no_outlier_df, columns = ['Education', 'Marital_Status'], drop_first=True)
```

Selection of Most useful Features

```
from sklearn.ensemble import RandomForestClassifier

X = no_outlier_df.drop(['Response'], axis=1).copy()
y = no_outlier_df['Response']
feature_names = X.columns

rf = RandomForestClassifier()
rf.fit(X, y)

importances = rf.feature_importances_
sorted_indices = np.argsort(importances)[::-1]
sorted_importances = importances[sorted_indices]
sorted_features = [feature_names[i] for i in sorted_indices]
cumulative_importance = np.cumsum(sorted_importances)
threshold = 0.95
num_features = np.argmax(cumulative_importance >= threshold) + 1
selected_features = sorted_features[:num_features]

selected_features
```

Selected features: ['Recency', 'days_since_customer', 'total_spent', 'Income', 'MntMeatProducts', 'MntWines', 'MntGoldProds', 'age', 'MntSweetProducts', 'AcceptedCmp3', 'AcceptedCmp5', 'MntFishProducts', 'NumStorePurchases', 'NumCatalogPurchases', 'MntFruits', 'AcceptedCmp1', 'NumWebVisitsMonth', 'NumDealsPurchases', 'NumWebPurchases', 'n_kids', 'Marital_Status_Single', 'Education_PhD', 'Marital_Status_Married', 'AcceptedCmp2', 'Teenhome', 'Marital_Status_Together']

I will create a new dataframe only with these features:

```
new_df = no_outlier_df[selected_features]
```

5. Determine Ideal Customer Profile

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled = scaler.fit_transform(new_df)
```

Parameter Grid Search for Optimal Number of Clusters

```python
from sklearn.model_selection import ParameterGrid
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

```python
best_score = -1
best_params = None
k_values = list(range(3, 47, 2))

param_grid = {'n_clusters':k_values}

wcss = [] # witin cluster sum of squares
sil_score = [] # silhoutte score
```

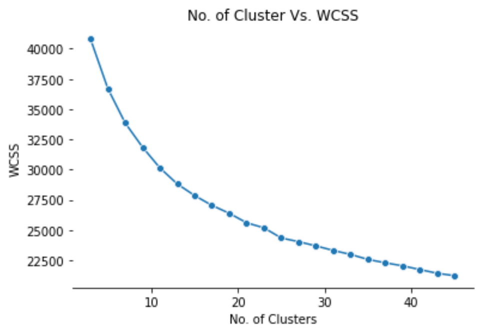
```python
for params in ParameterGrid(param_grid):
    model = KMeans(**params)
    labels = model.fit_predict(scaled)

    wcss.append(model.inertia_)

    score = silhouette_score(scaled, labels)
    sil_score.append(score)

    if score > best_score:
        best_score = score
        best_params = params
```

```python
print('Best Params: ', best_params, 'Best Silhouette Score: ', best_score)
```

```
Best Params:  {'n_clusters': 3} Best Silhouette Score:  0.16020091848338822
```

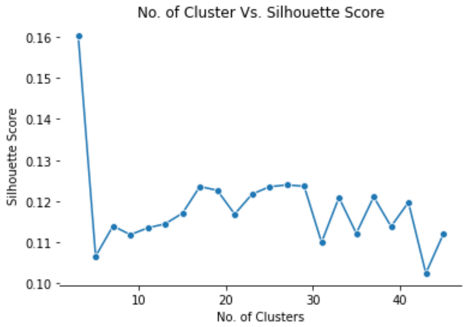
Number of Clusters vs. WCSS

```python
sns.lineplot(x=k_values, y=wcss, marker='o')
sns.despine(left=True)
plt.xlabel('No. of Clusters')
plt.ylabel('WCSS')
plt.title('No. of Cluster Vs. WCSS')
plt.show()
```



No. of Clusters vs. Silhouette Score

```
sns.lineplot(x=k_values, y=sil_score, marker='o')
sns.despine(left=True)
plt.xlabel('No. of Clusters')
plt.ylabel('Silhouette Score')
plt.title('No. of Cluster Vs. Silhouette Score')
plt.show()
```



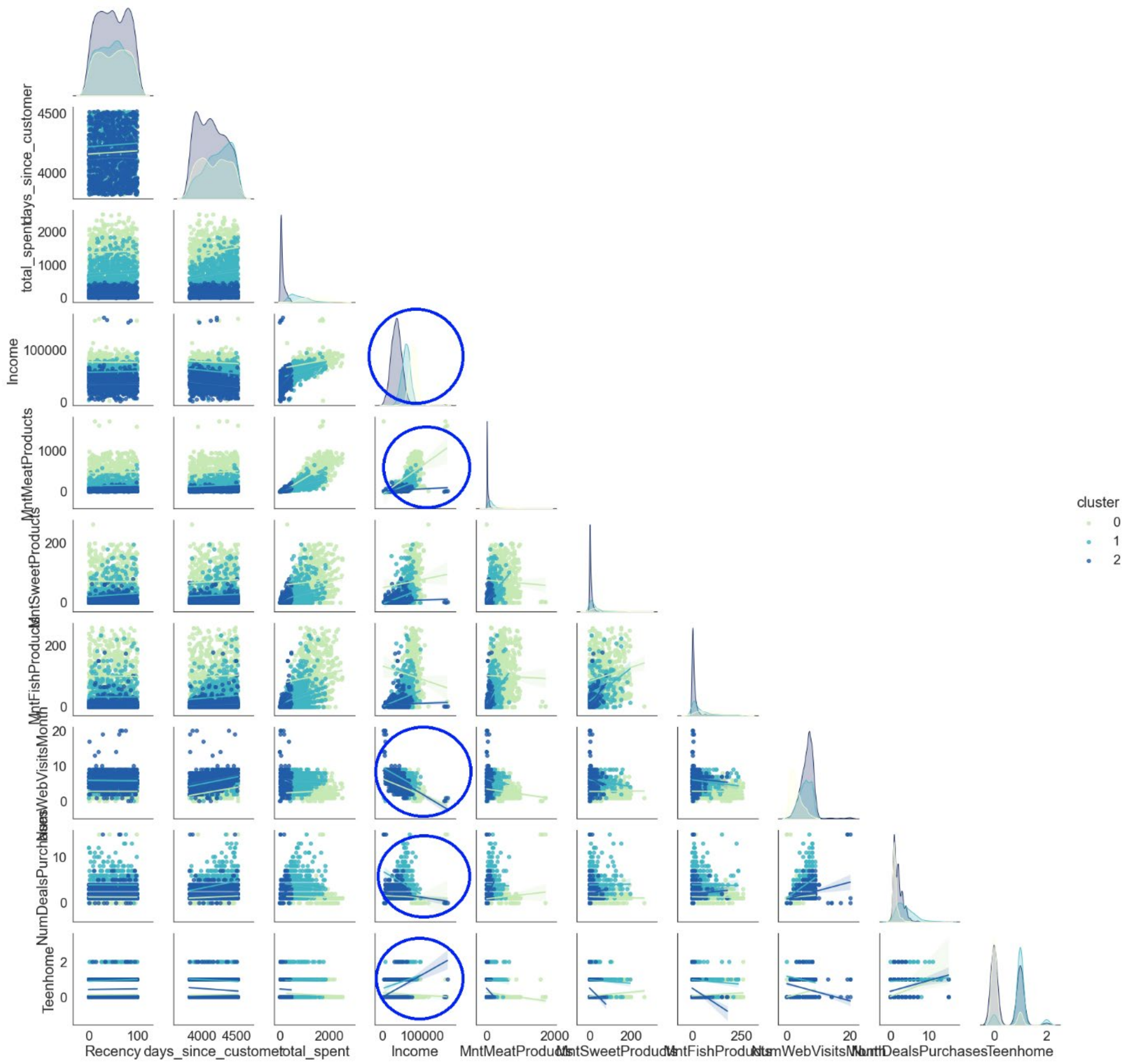
The WCSS (Within-Cluster Sum of Squares) did not reveal a clear elbow pattern. However, the Silhouette score was highest at 3 clusters. Therefore, for now, I will conclude that 3 clusters are ideal for this dataset. Let's proceed by rerunning the KMeans algorithm with 3 clusters as the hyperparameter.

```
model = KMeans(
    n_clusters = best_params['n_clusters'],
    random_state = 42,
    )

new_df['cluster'] = model.fit_predict(scaled)
```

For each feature vector, the associated cluster is assigned and saved as a new column ***cluster***.

Let's try to find how features relate to each other with respect to their assigned cluster.

```
sns.pairplot(new_df[features], hue='cluster', palette='viridis')
plt.show()
```



Observations:

- The cluster 0 contains the high-income customers.
- Strong positive correlation between income and amount spent on meat products
- Strong negative correlation between income and purchases from the web
- Strong negative correlation between income and number of deal purchases
- Negative correlation between income and number of teens at home for cluster 0 customers, and strong positive correlation for cluster 1 and 2 customers

Intuitively it makes sense the best performing customers are the ones who have high income, and spending the most on meat products, have fewer teens at their home, less frequent buyer from the web.

Let’s evaluate how many customers are in cluster 0 and their spending on the store.

```
new_df.groupby(by='cluster')[['Income', 'total_spent', 'MntMeatProducts']].describe().T
```

	cluster	0	1	2
Income	count	559.000000	608.000000	1048.000000
	mean	76524.393560	58192.106908	35262.696565
	std	11881.489498	10937.934989	14542.802818
	min	2447.000000	4428.000000	1730.000000
	25%	70293.500000	51508.250000	26085.000000
	50%	76653.000000	58554.000000	34733.000000
	75%	82347.000000	65337.750000	43018.500000
	max	160803.000000	94871.000000	162397.000000
total_spent	count	559.000000	608.000000	1048.000000
	mean	1393.747764	754.457237	102.482824
	std	417.608714	344.727471	94.765461
	min	277.000000	227.000000	5.000000
	25%	1081.500000	470.000000	38.750000
	50%	1370.000000	688.500000	65.000000
	75%	1676.500000	990.500000	137.000000
	max	2525.000000	1829.000000	467.000000
MntMeatProducts	count	559.000000	608.000000	1048.000000
	mean	461.774597	142.445724	24.146947
	std	248.072995	99.349749	26.628352
	min	3.000000	12.000000	0.000000
	25%	272.500000	70.000000	8.000000
	50%	424.000000	118.500000	15.000000
	75%	605.000000	184.000000	29.000000
	max	1725.000000	650.000000	217.000000

It is evident that Cluster 0 comprises a smaller number of customers, but they exhibit significantly higher spending at the store. Additionally, their average spending on meat products is substantially higher compared to the other two clusters.

I am now focusing exclusively on Cluster 0 and will disregard the other clusters (1 and 2). To determine an ideal customer profile, I will use a tree induction model to generate cluster descriptions.

To achieve this, I will transform first the cluster labels into a binary classification problem, where Cluster 0 will be assigned a binary class of 1, and Clusters 1 and 2 will be assigned a binary class of 0. The newly created binary cluster feature will be used as the target variable for model development.

```
new_df['cluster'] = new_df['cluster'].map({0:1, 1:0, 2:0})
```

Further, I will employ a DecisionTreeClassifier to construct the induction tree, which will help identify the key features that differentiate Cluster 0 from the other clusters.

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
```

```
X = new_df.drop(['cluster'], axis=1)
y = new_df['cluster']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=24)

classifier = DecisionTreeClassifier(ccp_alpha=0.01)
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)
```

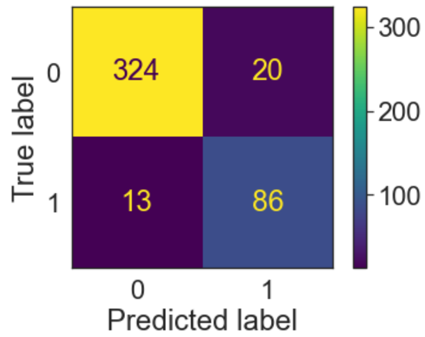
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       344
           1       0.81      0.87      0.84        99

    accuracy                           0.93       443
   macro avg       0.89      0.91      0.90       443
weighted avg       0.93      0.93      0.93       443
```

Confusion matrix:

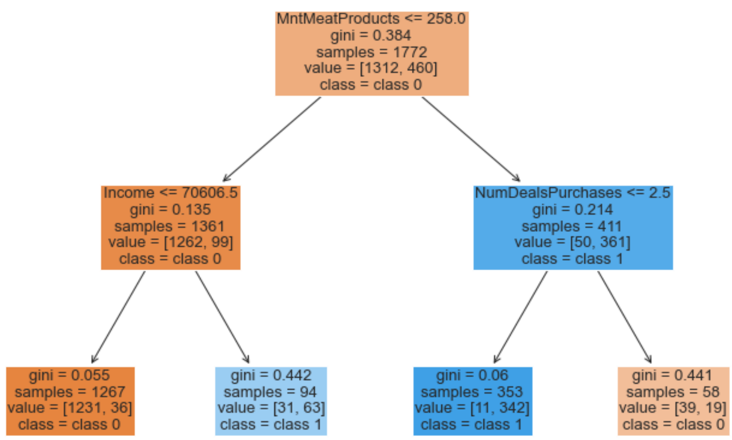
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1cd742082b0>
```



Let’s export a tree to see how tree induction is developed on different features.

```
from sklearn.tree import export_text
plt.figure(figsize=(12, 8))
plot_tree(classifier, feature_names=list(X.columns), class_names=["class 0", "class 1"], filled=True)
plt.show()
```



As observed from the decision tree, the amount spent on meat products serves as the best feature to split the data based on entropy. On the right branch, the key distinguishing feature is the number of deal purchases, while on the left branch, the split is driven by income.

These insights make it significantly easier to develop an ideal customer profile for Cluster 0, focusing on high spenders on meat products, frequent deal purchasers, and customers with higher income levels.

Ideal Customer Profile:

- **Customers who have spent more than $258 on meat products and have made 2 or fewer deal purchases.**

OR

- **Customers with an income greater than $70,606.**

These criteria highlight two distinct groups of ideal customers in Cluster 0 — high meat spenders with limited deal usage or high-income customers, both of which are valuable for targeted marketing or personalized campaigns.

7. Post-Project Review

Areas for improvement:

- **Outlier Treatment:** Outliers in six features were not removed, leaving their impact on the results uncertain. Further analysis can be conducted by removing these outliers and reassessing cluster quality.
- **Model Performance:** The confusion matrix for the DecisionTreeClassifier indicates the presence of false positives and false negatives. There is an opportunity to refine the model by tuning its hyperparameters and re-evaluating its performance using the confusion matrix.
- **Business Relevance:** Is there a difference between profitable customers and the average customer? Utilization of statistical tests is required to confirm or disconfirm it.