Projeto de Disciplina - Validação de modelos de clusterização [22E4_3]

Nathalia de Almeida Castelo Branco

Etapa 01 - Infraestrutura

In [ ]:
```
# Importação das dependências

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

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.decomposition import PCA
import scipy.cluster.hierarchy as sch
from sklearn.neighbors import NearestNeighbors

from sklearn.metrics import pairwise_distances_argmin_min

from yellowbrick.cluster import KElbowVisualizer
from yellowbrick.cluster import SilhouetteVisualizer

from ds_utils.unsupervised import plot_cluster_cardinality, plot_cluster_magnitude,
from scipy.spatial.distance import euclidean

from DBCV import DBCV

sns.set(style='whitegrid')
```

In [ ]:
```
def plot_cluster_points(df, labels, ax=None, hue="cluster", legend="auto"):
    pca = PCA(2)
    pca_data = pd.DataFrame(pca.fit_transform(df), columns=['PC1','PC2'])
    pca_data['cluster'] = pd.Categorical(labels)
    sns.scatterplot(x="PC1", y="PC2", hue=hue, data=pca_data, ax=ax, legend=legend)
```

Etapa 02 - Escolha da base de dados

1. Escolha uma base de dados para realizar o trabalho. Essa base será usada em um problema de clusterização.

Dicionário de Dados

| **FEATURES - CLIENTES** | DESCRIÇÃO | |---|---| | ID | Identificador único do cliente | | Year_Birth | Ano de aniversário do cliente | | Education | Nível de educação do cliente | | Marital_Status | Estado civil do cliente | | Income | Renda anual do cliente | | Kidhome | Número de crianças na casa do cliente | | Teenhome | Número de adolescentes | |

Dt_Customer | Data em que se tornou cliente | | Recency | Número de dias desde a última compra | | Complain | Reclamação: se reclamou nos últimos dois anos, 1. Caso contrário, 0. |

| **FEATURES - PRODUTOS** | DESCRIÇÃO | |---|---| | MntWines | Total gasto em vinho, últimos dois anos | | MntFruits | Total gasto em frutas, últimos dois anos | | MntMeatProducts | Total gasto em carnes ou alimentos, últimos dois anos | | MntFishProducts | Total gasto em peixes, últimos dois anos | | MntSweetProducts | Total gasto em doces, últimos dois anos | | MntGoldProds | Total gasto em produtos premium, últimos dois anos |

| **FEATURES - OFERTAS** | DESCRIÇÃO | |---|---| | NumDealsPurchases | Número de pedidos feitos com desconto | | AcceptedCmp1 | Aceite de oferta: se o cliente aceitou a oferta na primeira campanha, 1. Caso contrário, 0. | | AcceptedCmp2 | Aceite de oferta: se o cliente aceitou a oferta na segunda campanha, 1. Caso contrário, 0. | | AcceptedCmp3 | Aceite de oferta: se o cliente aceitou a oferta na terceira campanha, 1. Caso contrário, 0. | | AcceptedCmp4 | Aceite de oferta: se o cliente aceitou a oferta na quarta campanha, 1. Caso contrário, 0. | | AcceptedCmp5 | Aceite de oferta: se o cliente aceitou a oferta na quinta campanha, 1. Caso contrário, 0. | | Response | Aceite de oferta: se o cliente aceitou a oferta na última campanha, 1. Caso contrário, 0. |

| **FEATURES - LOCAL DA COMPRA** | DESCRIÇÃO | |---|---| | NumWebPurchases | Número de compras feitas pelo website | | NumCatalogPurchases | Número de compras feitas usando um catálogo | | NumStorePurchases | Número de compras feitas presencialmente na loja | | NumWebVisitsMonth | Número de visitas ao website da empresa no último mês |

As colunas Z_CostContact e Z_Revenue não aparentam possuir significado, serão excluídas do estudo

In [ ]:
```
data = pd.read_csv('data\marketing_campaign.csv', sep=",")
data.head()
```

Out[ ]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	

5 rows × 29 columns

In [ ]:
```
data.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
```

In [ ]:
```
print(data.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
dtype: int64
```

Nota-se que a feature da renda do cliente possui 24 valores nulos. Estes valores serão removidos.

2. Escreva a justificativa para a escolha de dados, dando sua motivação e objetivos.

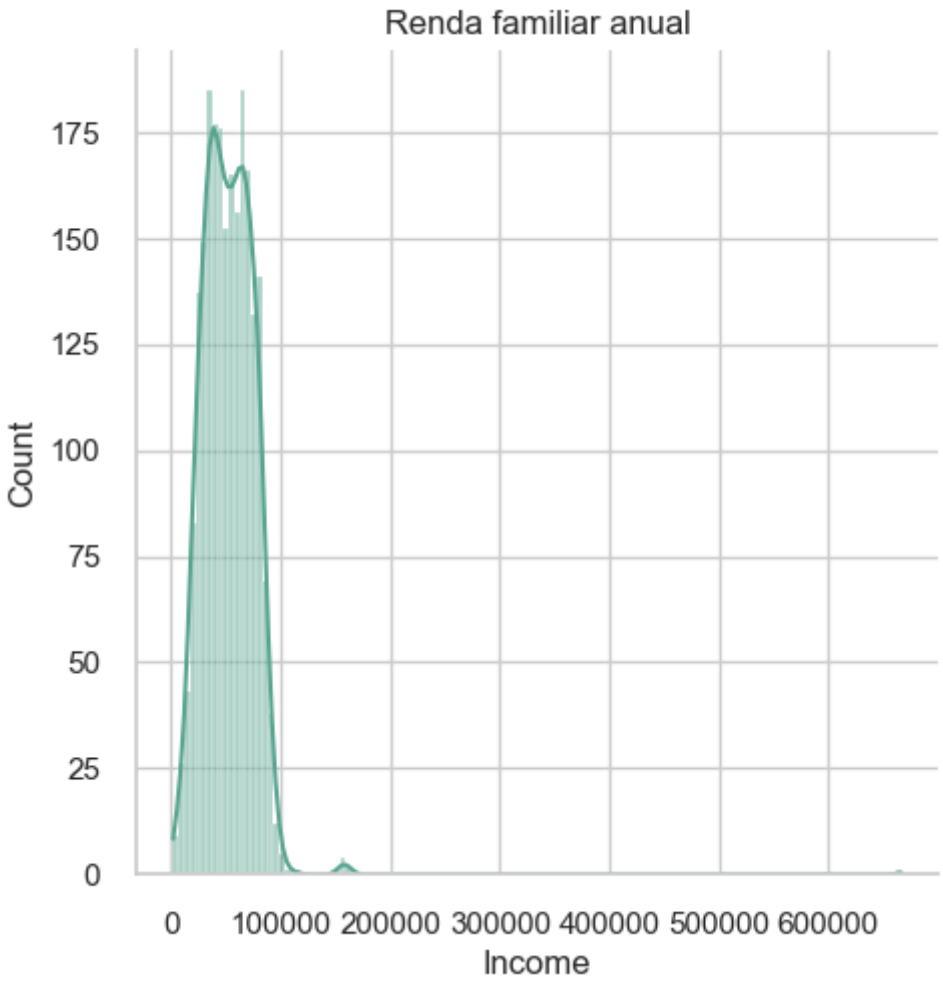
> Esta base de dados foi escolhida com o objetivo de praticar a segmentação de clientes, visto que no momento trabalho na interface entre tecnologia e marketing. A motivação para escolha se baseia em treinar a metodologia de análise de segmentos de clientes e aplicar em um case na empresa em que trabalho.

Etapa 03 - Mostre através de gráficos a faixa dinâmica das variáveis que serão usadas nas tarefas de clusterização. Analise os resultados mostrados. O que deve ser feito com os dados antes da etapa de clusterização?

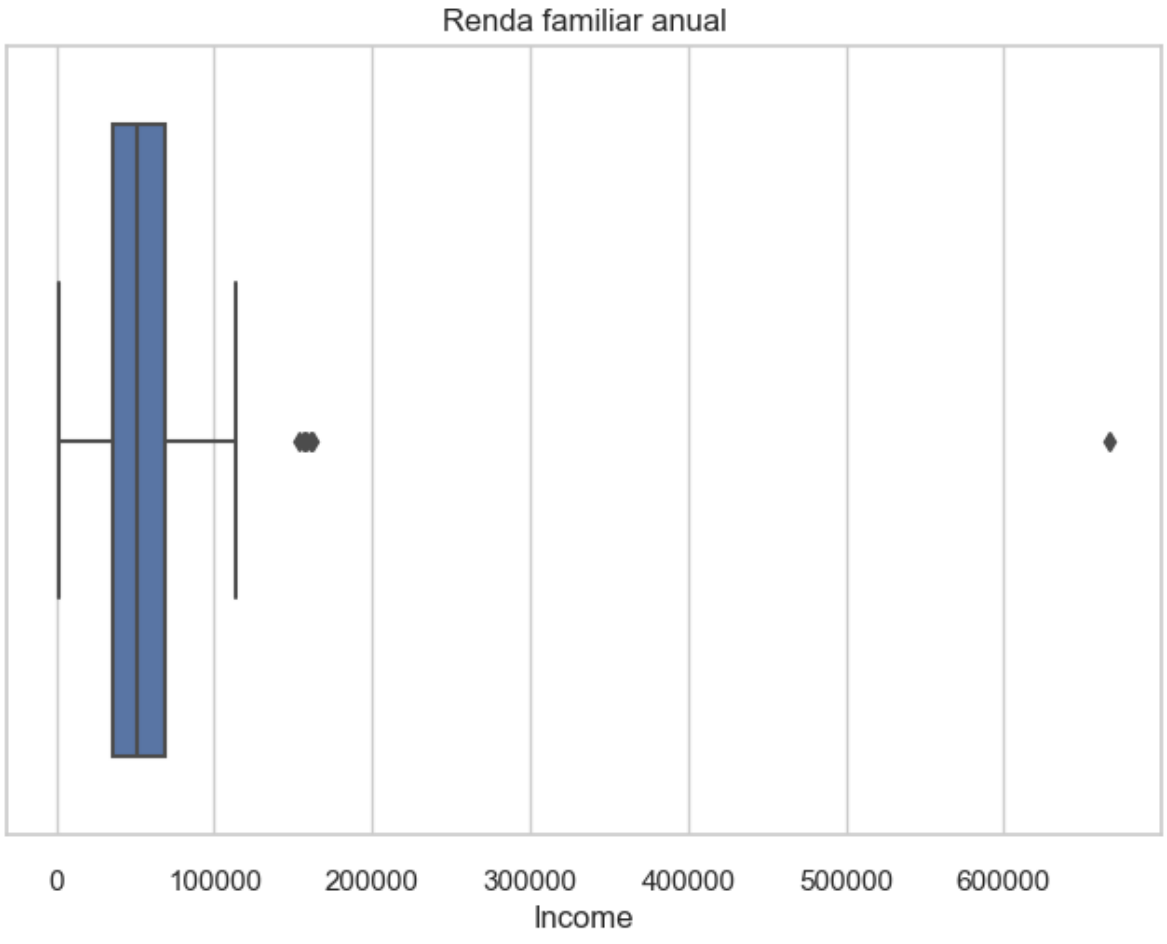
Demonstração da faixa dinâmica das variáveis

Renda familiar anual

In [ ]:
```
sns.displot(data=data, x = 'Income', color='#59A590', kde=True).set(title=f'Renda
```



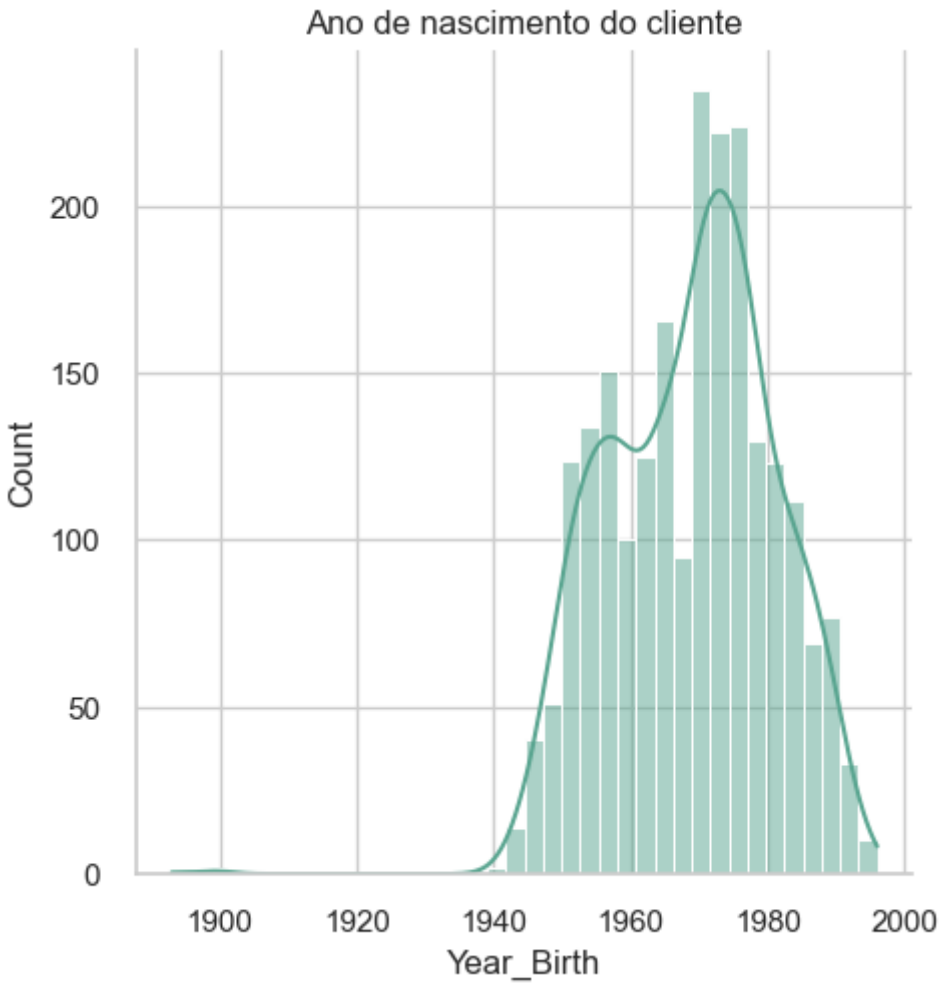
In [ ]:
```
sns.boxplot(x=data['Income']).set(title=f'Renda familiar anual');
```



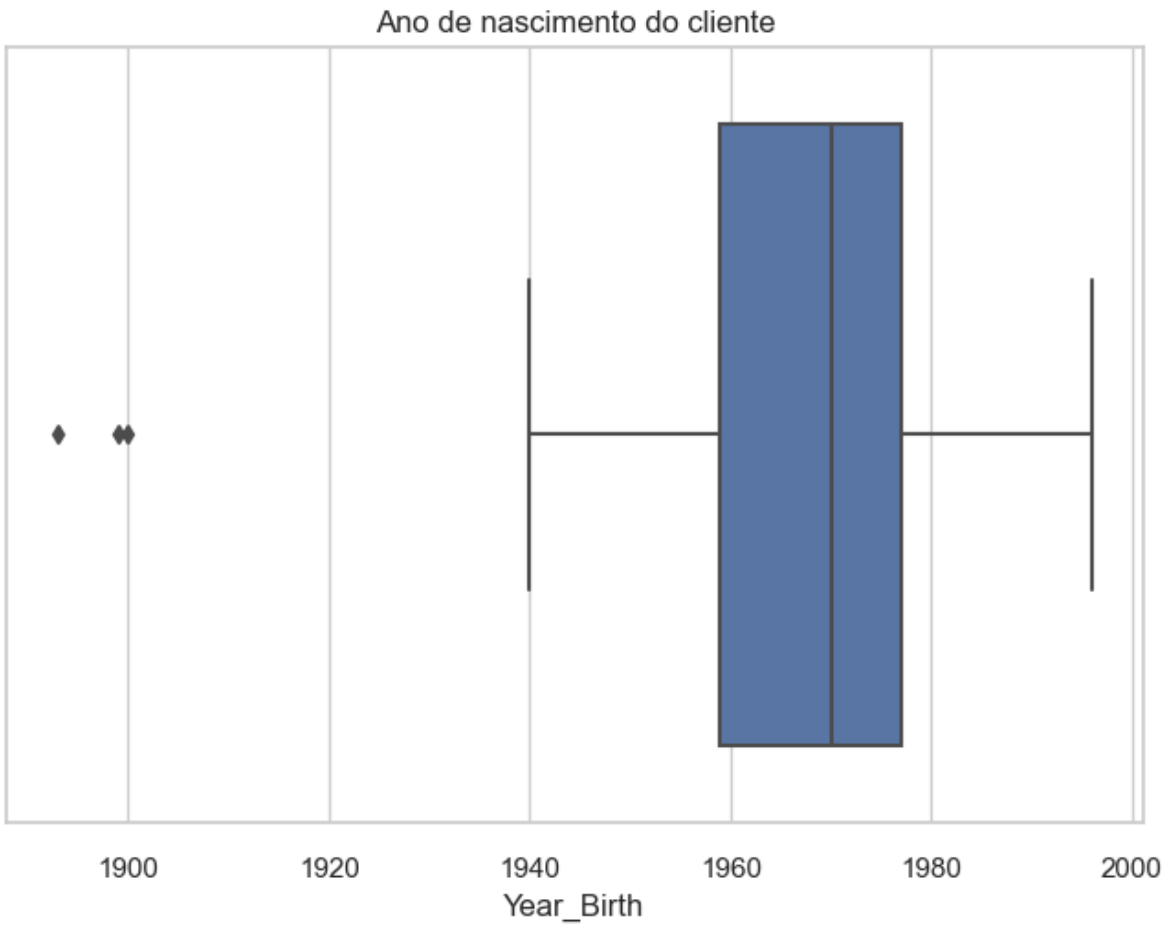
Para feature **Income**, os pontos próximos de 200 mil são aceitáveis. Já o ponto acima de 600 mil pode ser considerado um outlier.

Ano de nascimento

In [ ]:
```
sns.displot(data=data, x = 'Year_Birth', color='#59A590', kde=True).set(title=f'Ano
```



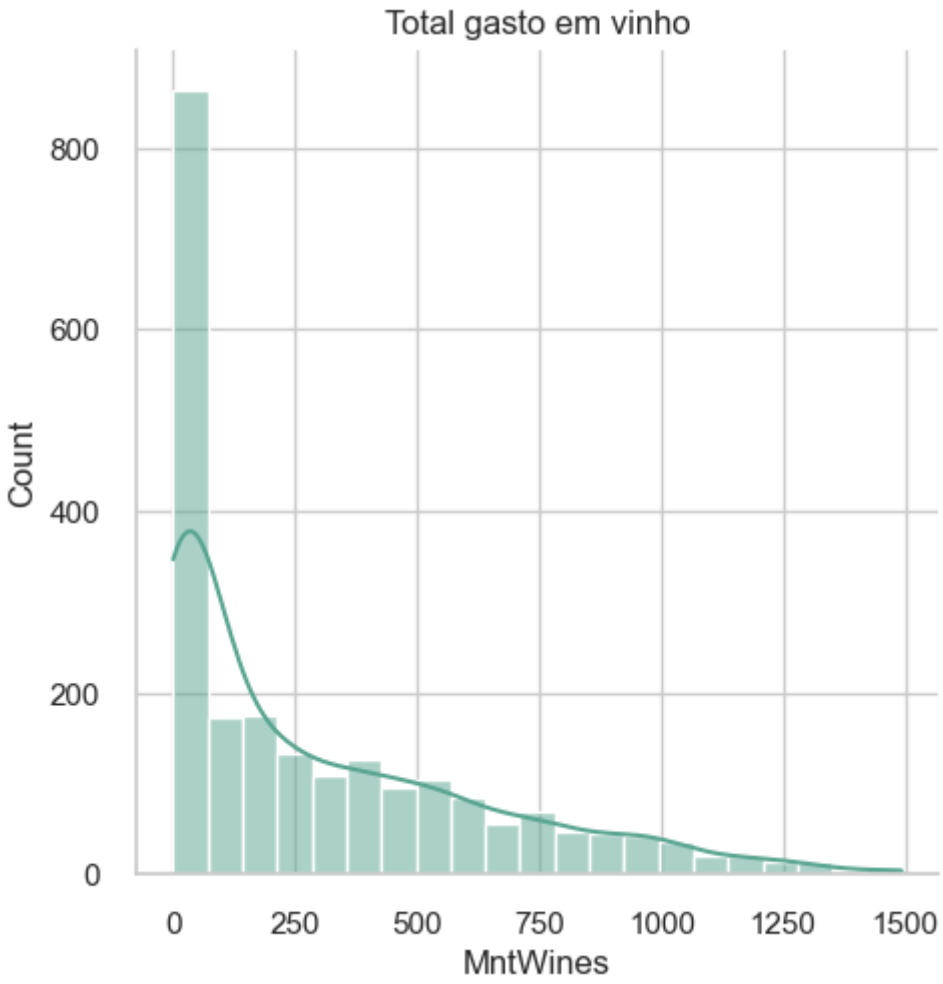
In [ ]:
```
sns.boxplot(x=data['Year_Birth']).set(title=f'Ano de nascimento do cliente');
```



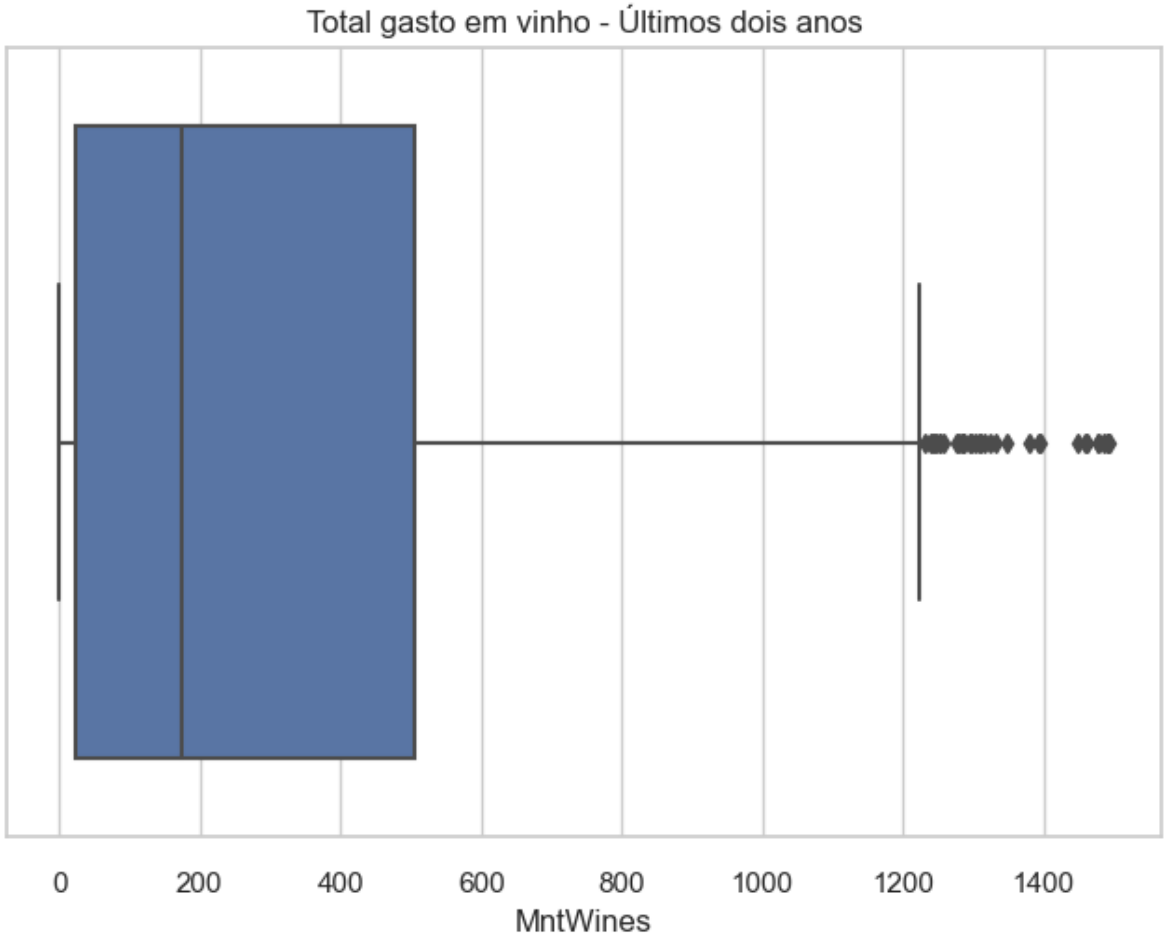
Para a feature **Year_Birth**, os anos antecedentes a 1940 podem ser considerados outliers, visto que, a partir de 1940, os clientes já possuem atualmente 83 anos.

Valor gasto em vinho nos últimos 2 anos

In [ ]:
```
sns.displot(data=data, x = 'MntWines', color='#59A590', kde=True).set(title=f'Total
```

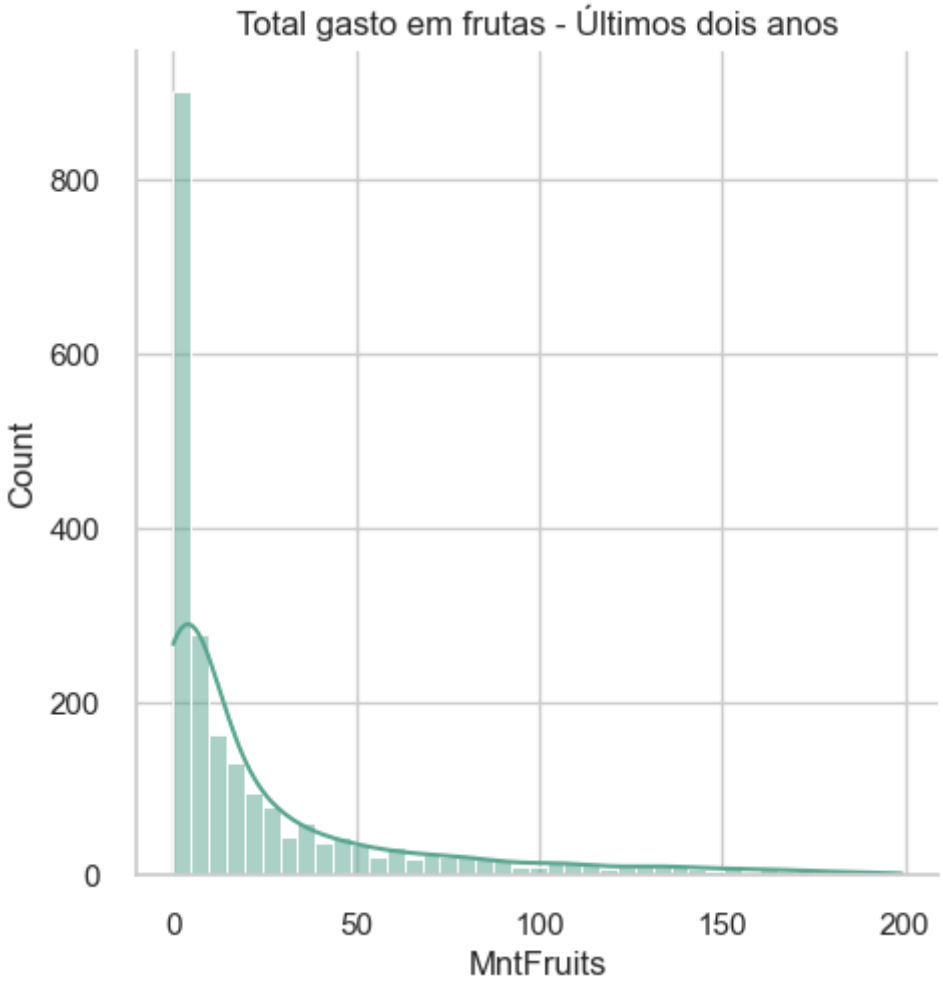


In [ ]:
```
sns.boxplot(x=data['MntWines']).set(title=f'Total gasto em vinho - Últimos dois anc
```

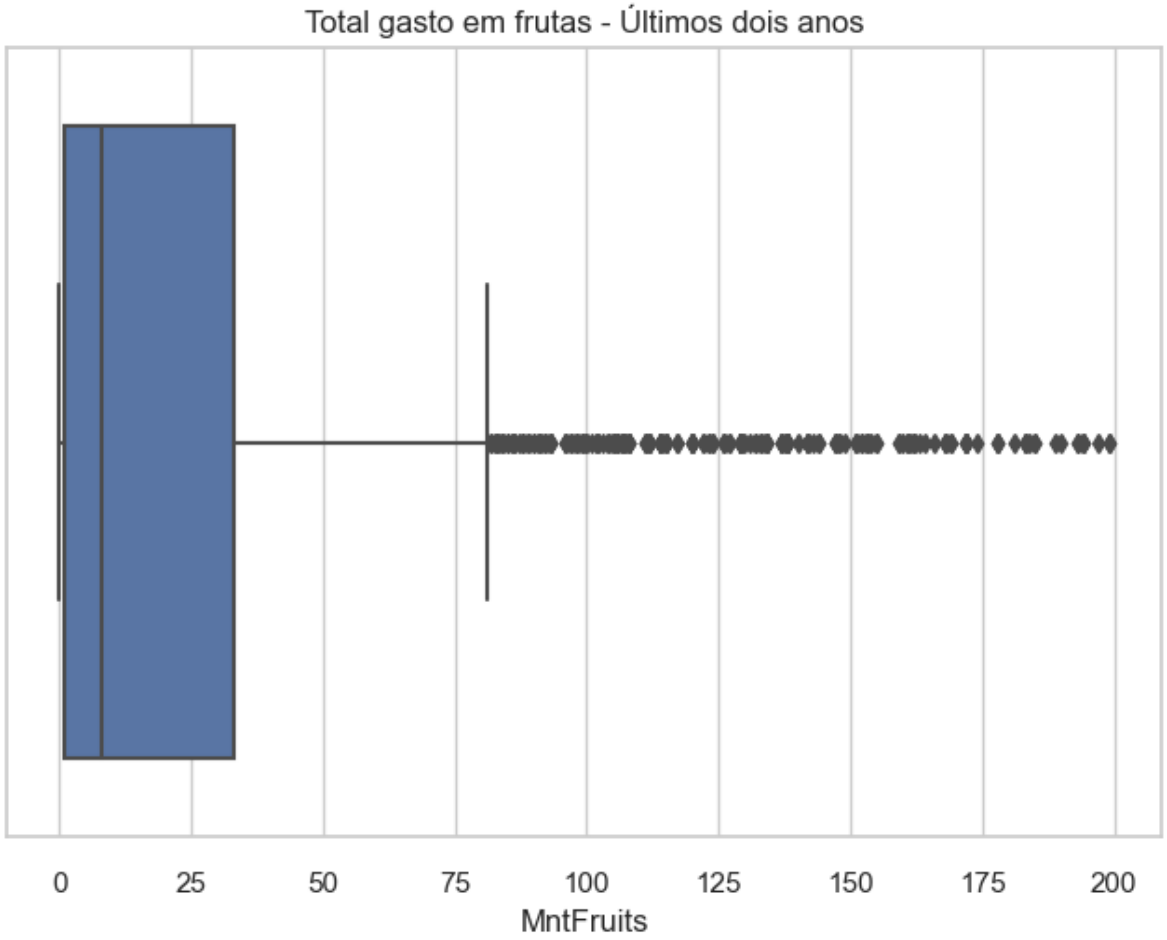


Valor gasto em frutas nos últimos 2 anos

In [ ]:
```
sns.displot(data=data, x = 'MntFruits', color='#59A590', kde=True).set(title=f'Tota
```

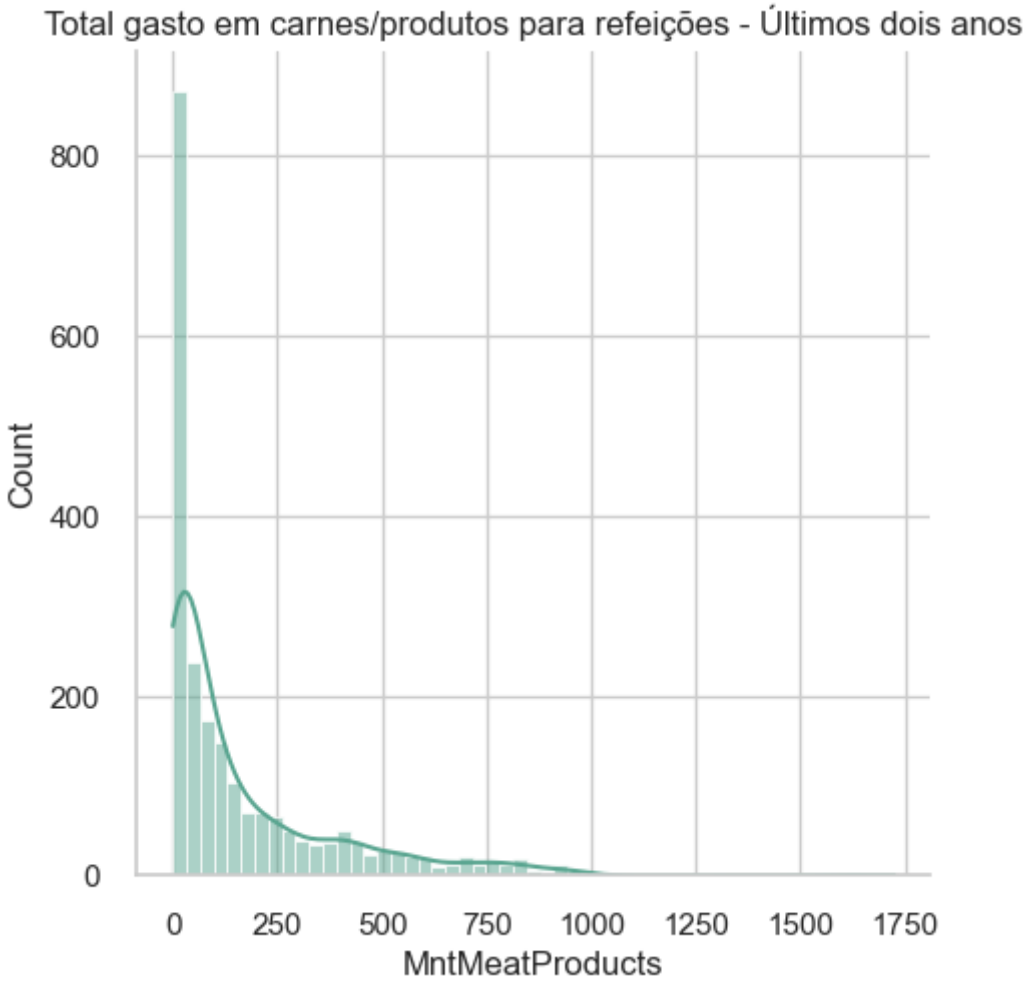


In [ ]:
```
sns.boxplot(x=data['MntFruits']).set(title=f'Total gasto em frutas - Últimos dois a
```

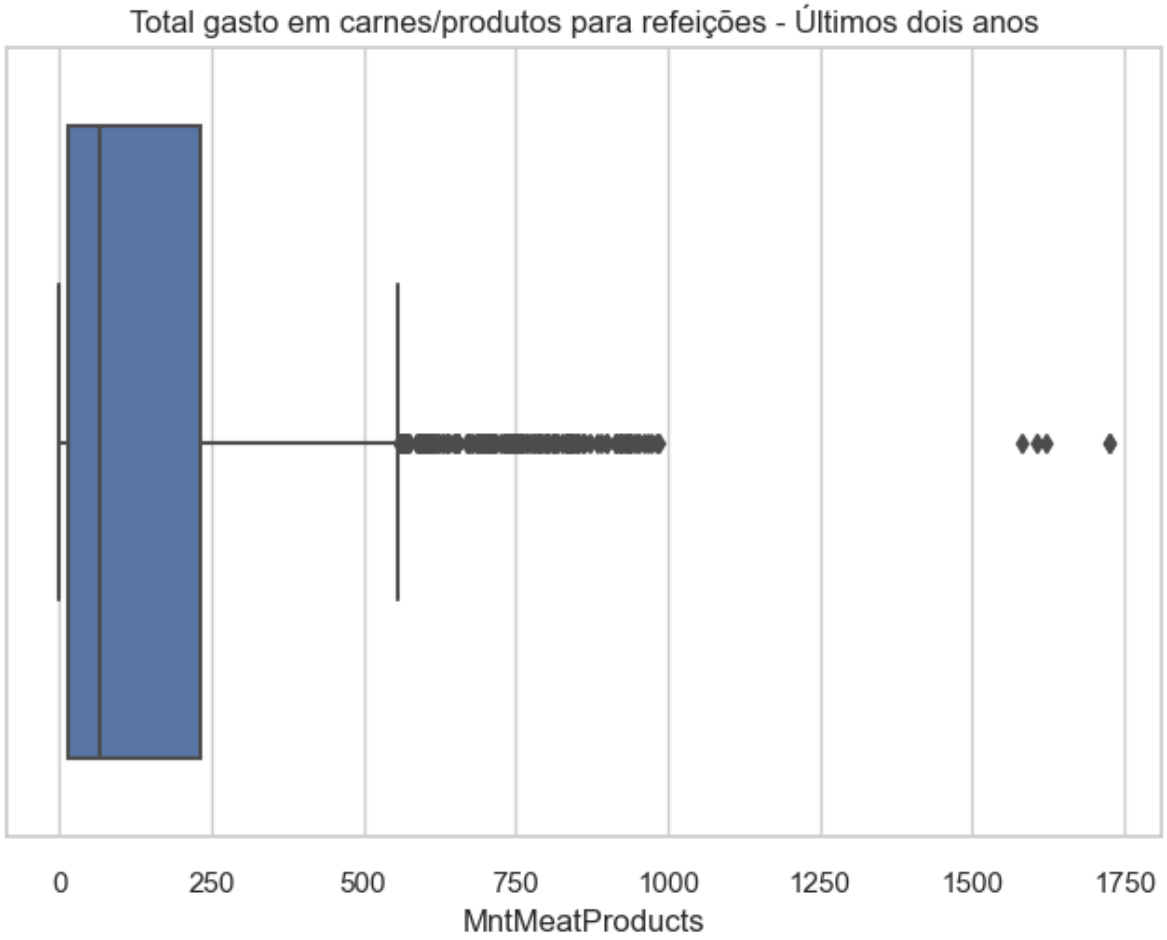


Valor gasto em carnes/produtos para refeições nos últimos 2 anos

In [ ]:
```
sns.displot(data=data, x = 'MntMeatProducts', color='#59A590', kde=True).set(title=
```

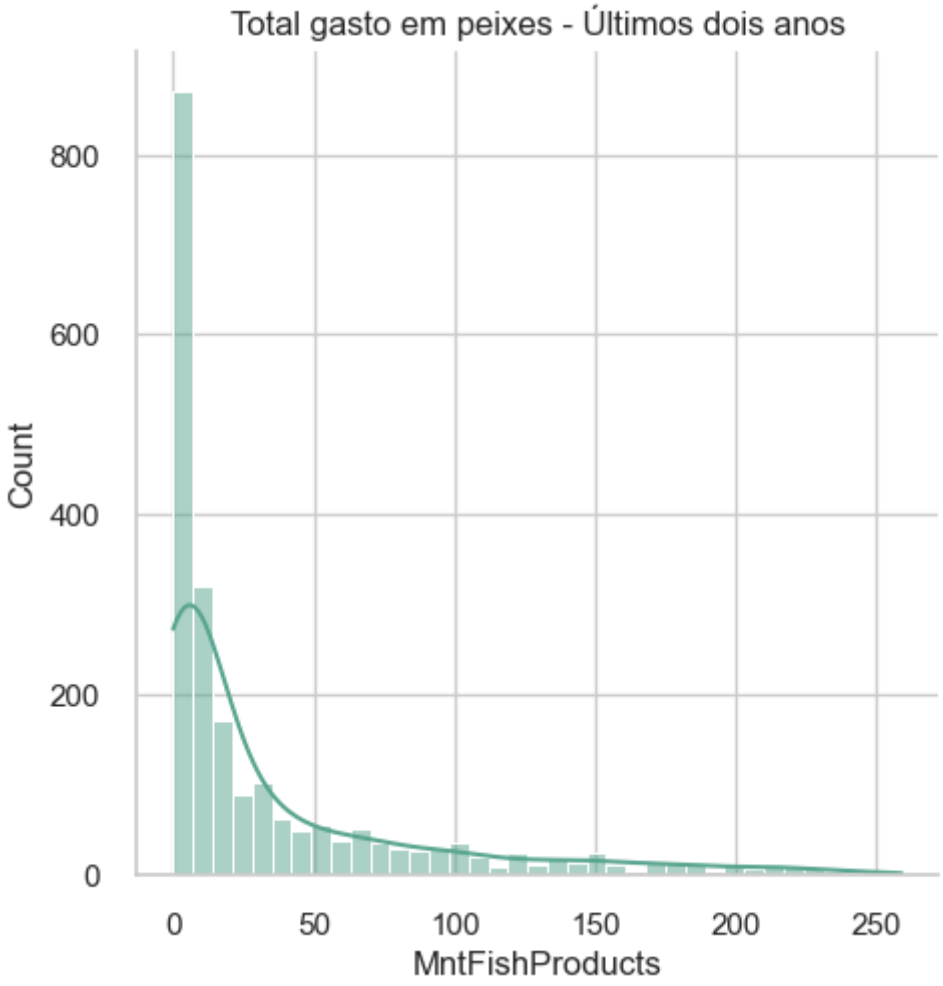


In [ ]:
```
sns.boxplot(x=data['MntMeatProducts']).set(title=f'Total gasto em carnes/produtos p
```

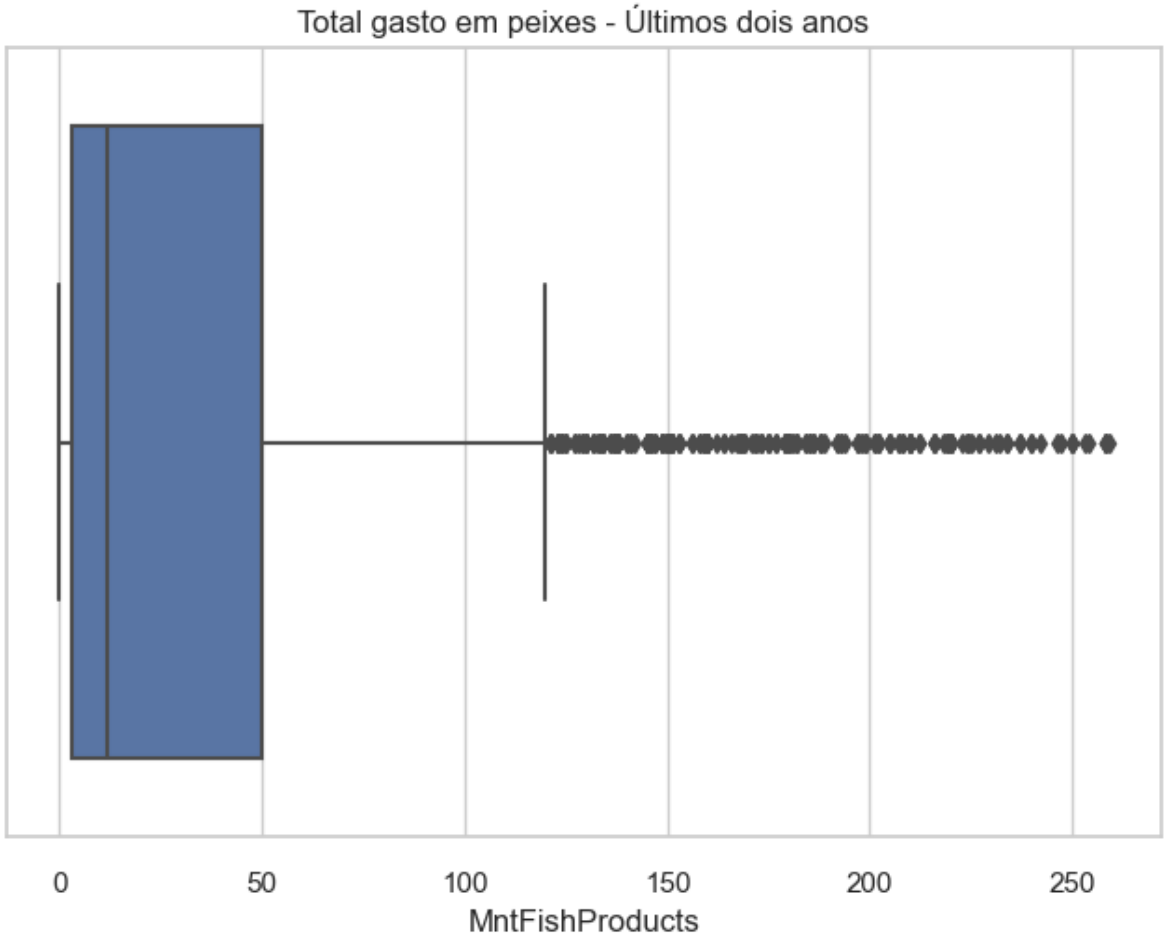


Valor gasto em peixes nos últimos 2 anos

In [ ]:
```
sns.displot(data=data, x = 'MntFishProducts', color='#59A590', kde=True).set(title=
```

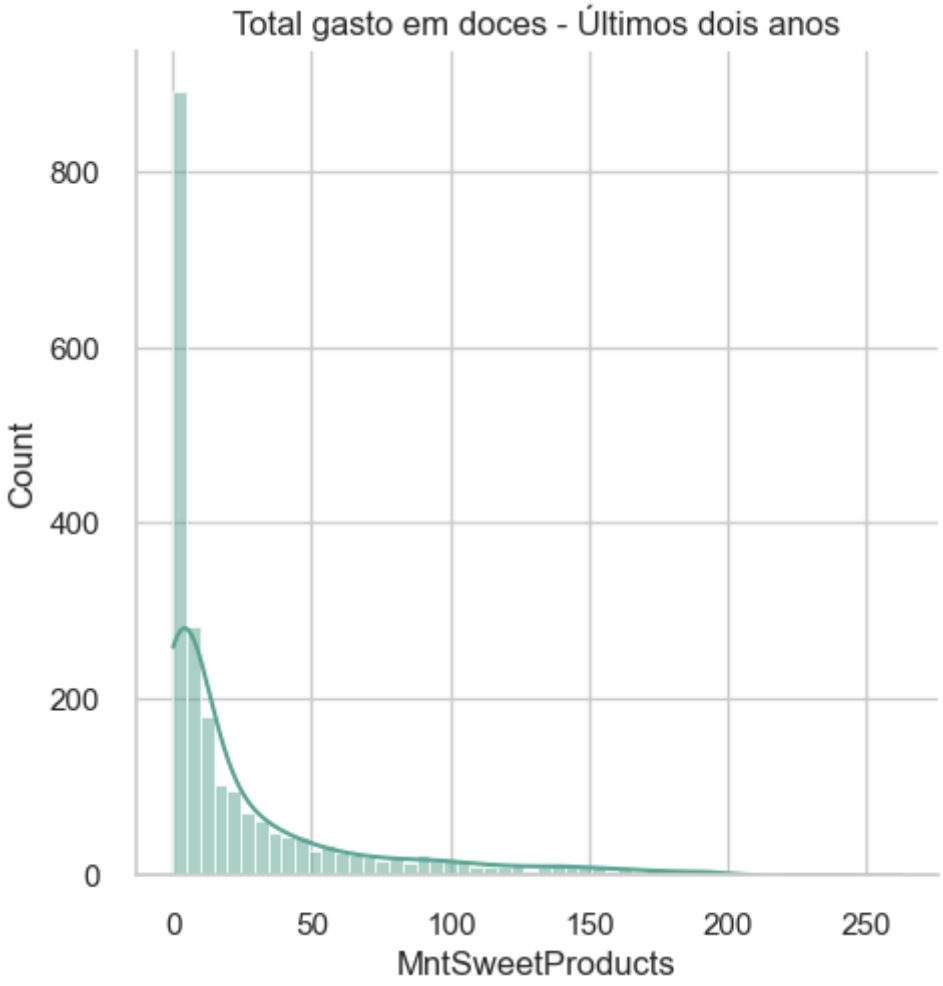


In [ ]:
```
sns.boxplot(x=data['MntFishProducts']).set(title=f'Total gasto em peixes - Últimos
```

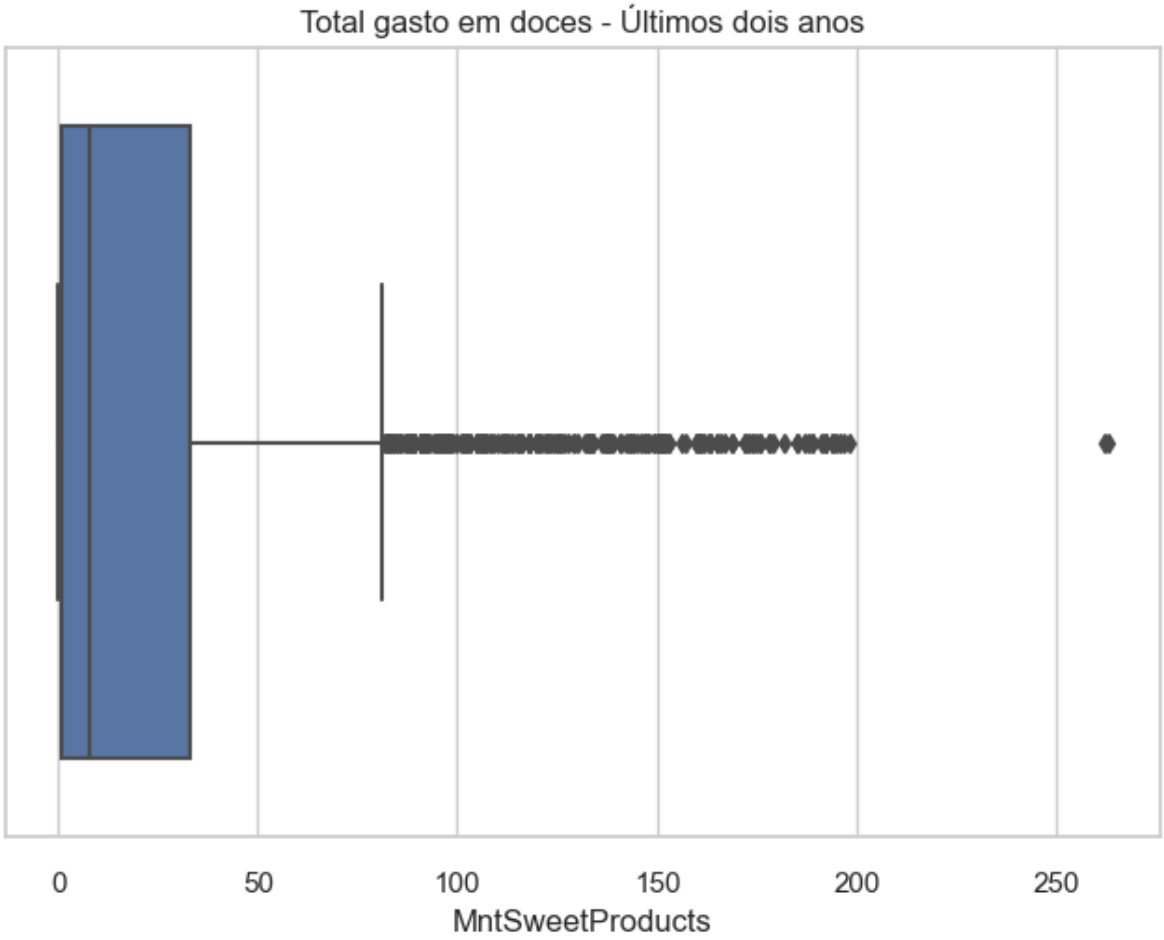


Valor gasto em doces nos últimos 2 anos

In [ ]:
```
sns.displot(data=data, x = 'MntSweetProducts', color='#59A590', kde=True).set(title
```

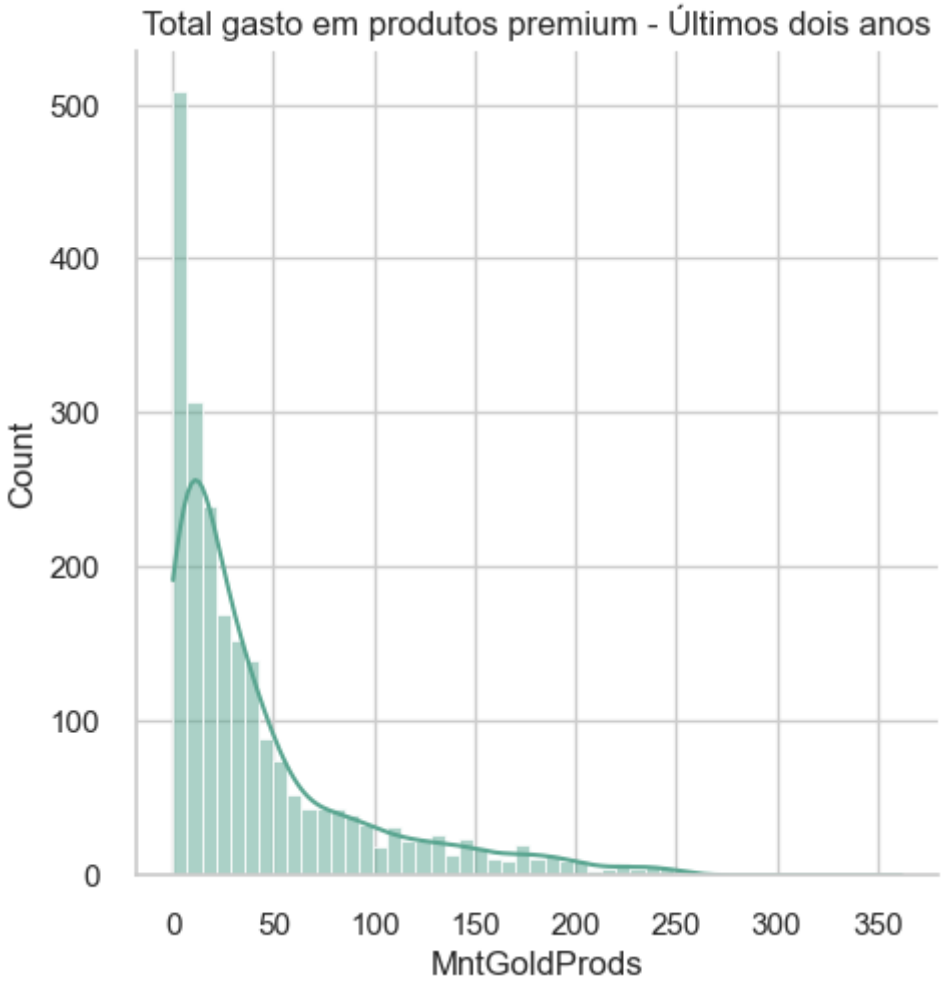


In [ ]:
```
sns.boxplot(x=data['MntSweetProducts']).set(title=f'Total gasto em doces - Últimos
```

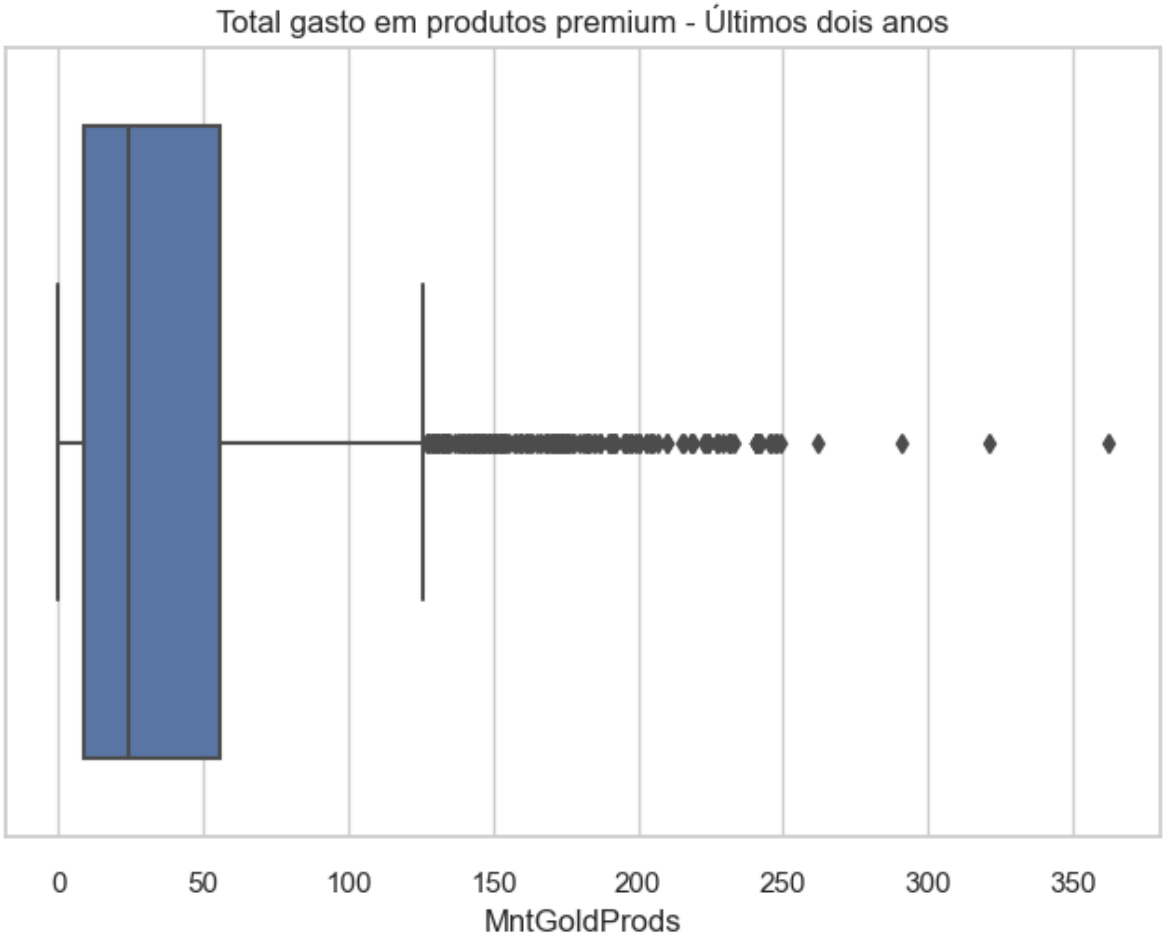


Valor gasto em produtos premium nos últimos 2 anos

In [ ]:
```
sns.displot(data=data, x = 'MntGoldProds', color='#59A590', kde=True).set(title=f'
```



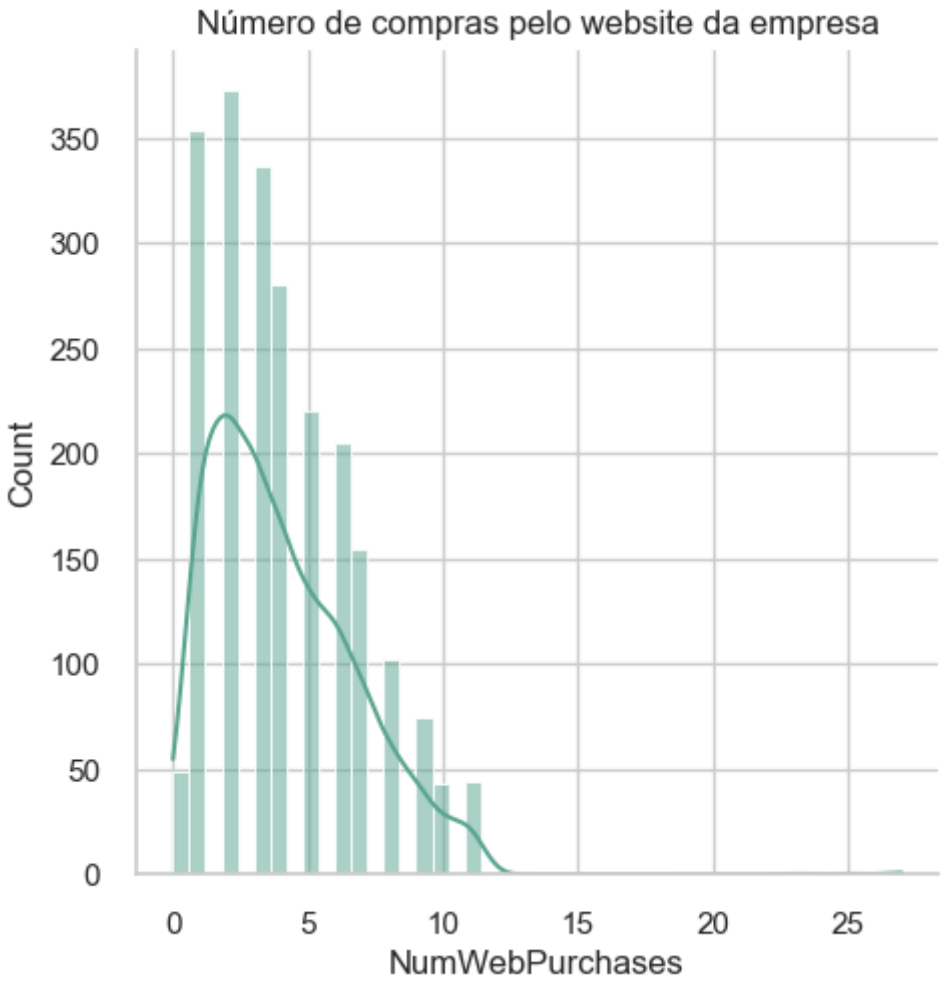
In [ ]:
```
sns.boxplot(x=data['MntGoldProds']).set(title=f'Total gasto em produtos premium - U
```



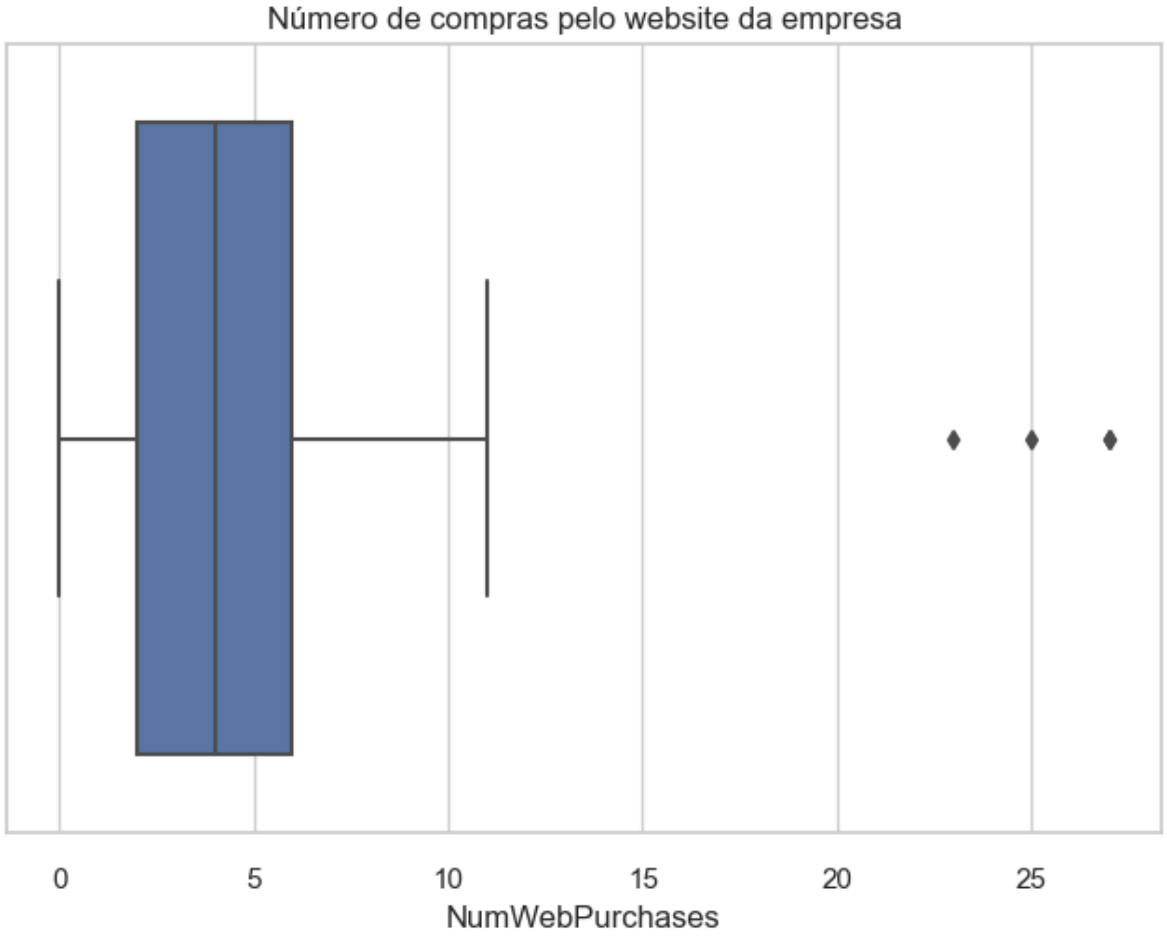
A distribuição desta variável é interessante pois é possível observar um comportamento de compra. Em geral, não há grande gasto com produtos premium, mas existe uma série de outliers que aparecem nesta distribuição, representando que existe um mercado para esse consumidor.

Número de compras pelo website da empresa

In [ ]:
```
sns.displot(data=data, x = 'NumWebPurchases', color='#59A590', kde=True).set(title=
```

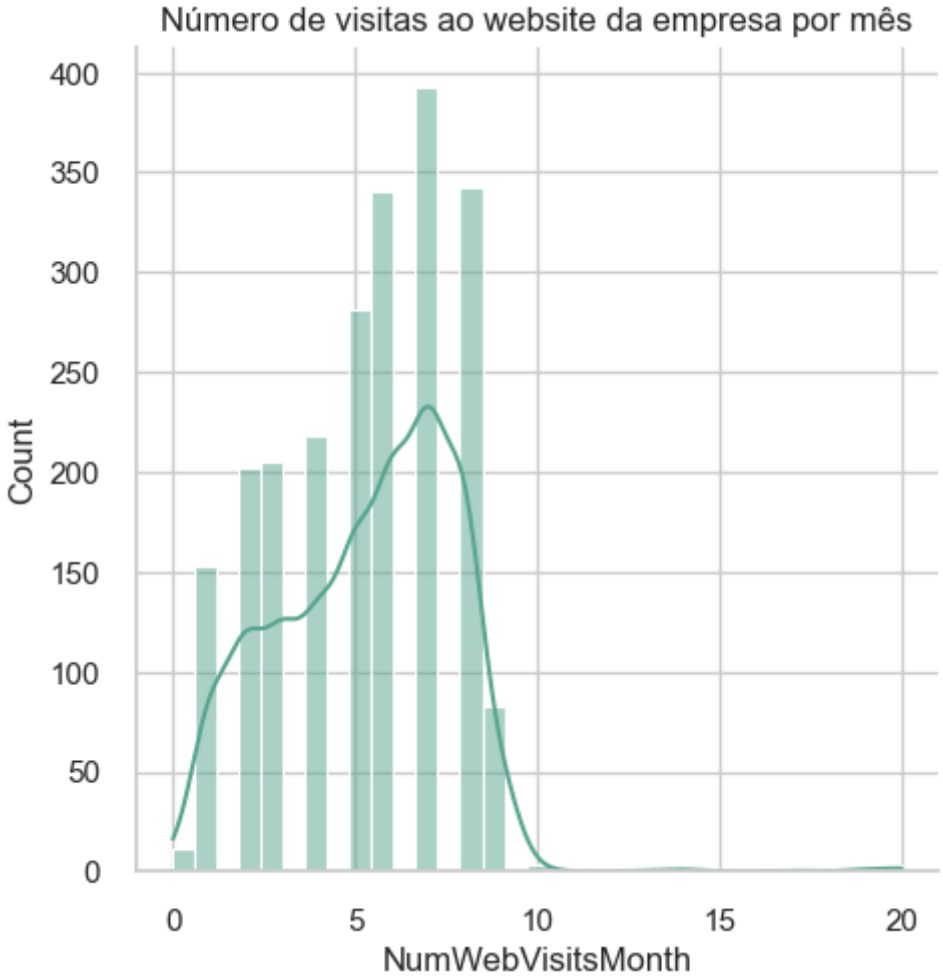


In [ ]:
```
sns.boxplot(x=data['NumWebPurchases']).set(title=f'Número de compras pelo website d
```

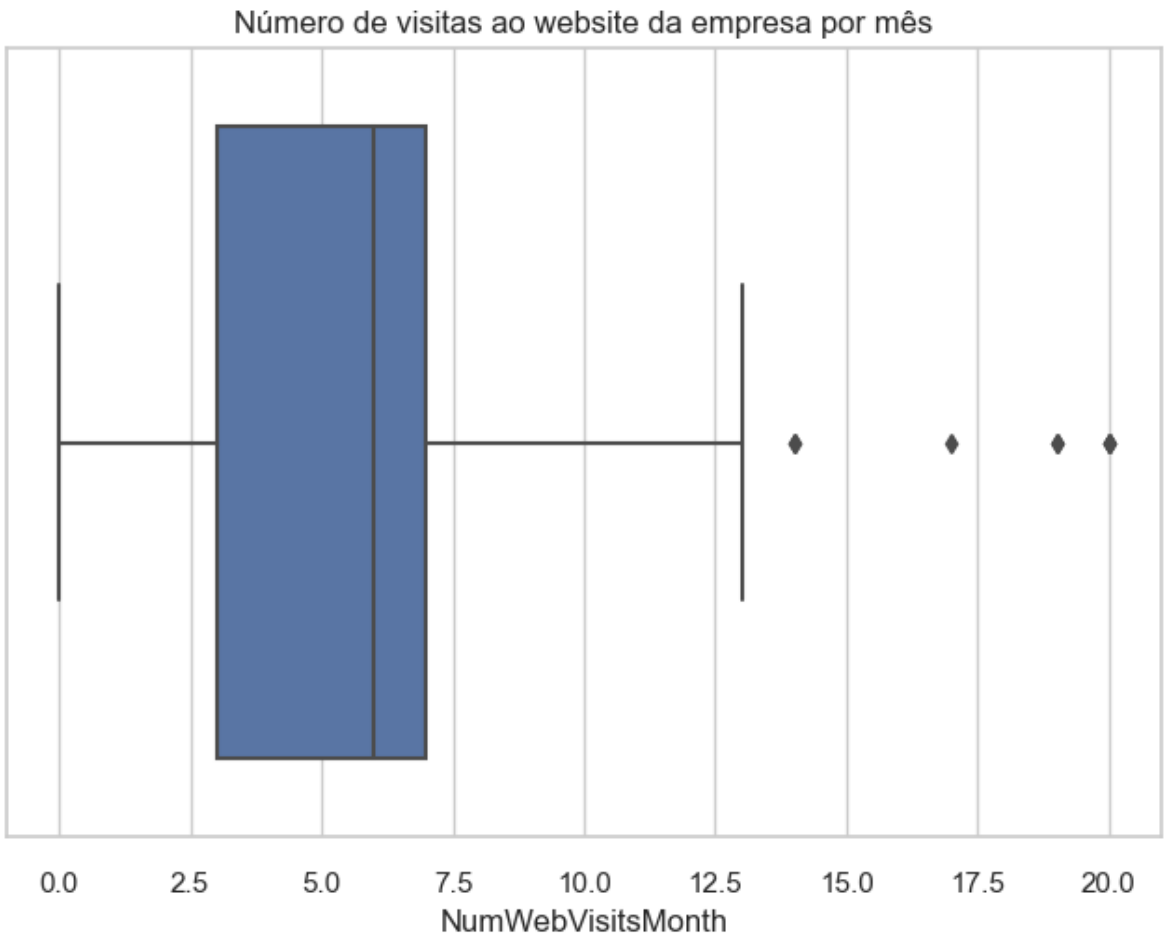


Número de visitas ao website da empresa por mês

In [ ]:
```
sns.displot(data=data, x = 'NumWebVisitsMonth', color='#59A590', kde=True).set(titl
```

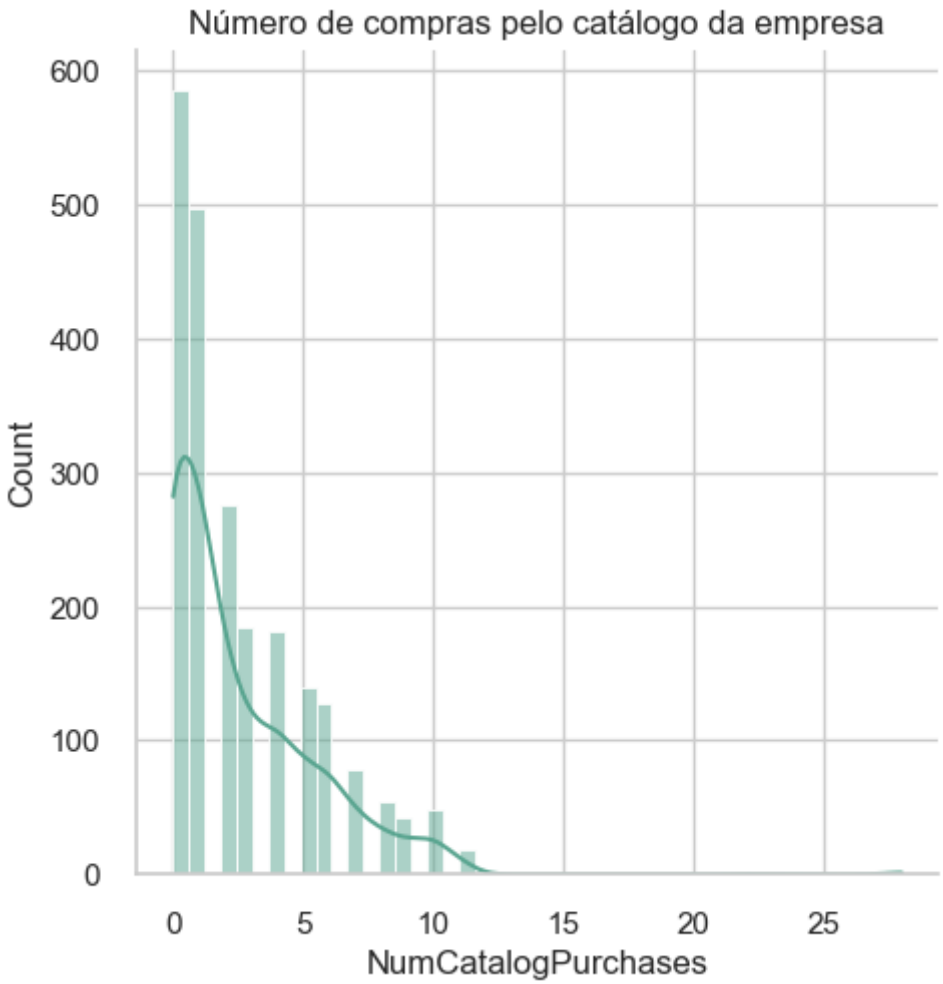


In [ ]:
```
sns.boxplot(x=data['NumWebVisitsMonth']).set(title=f'Número de visitas ao website d
```

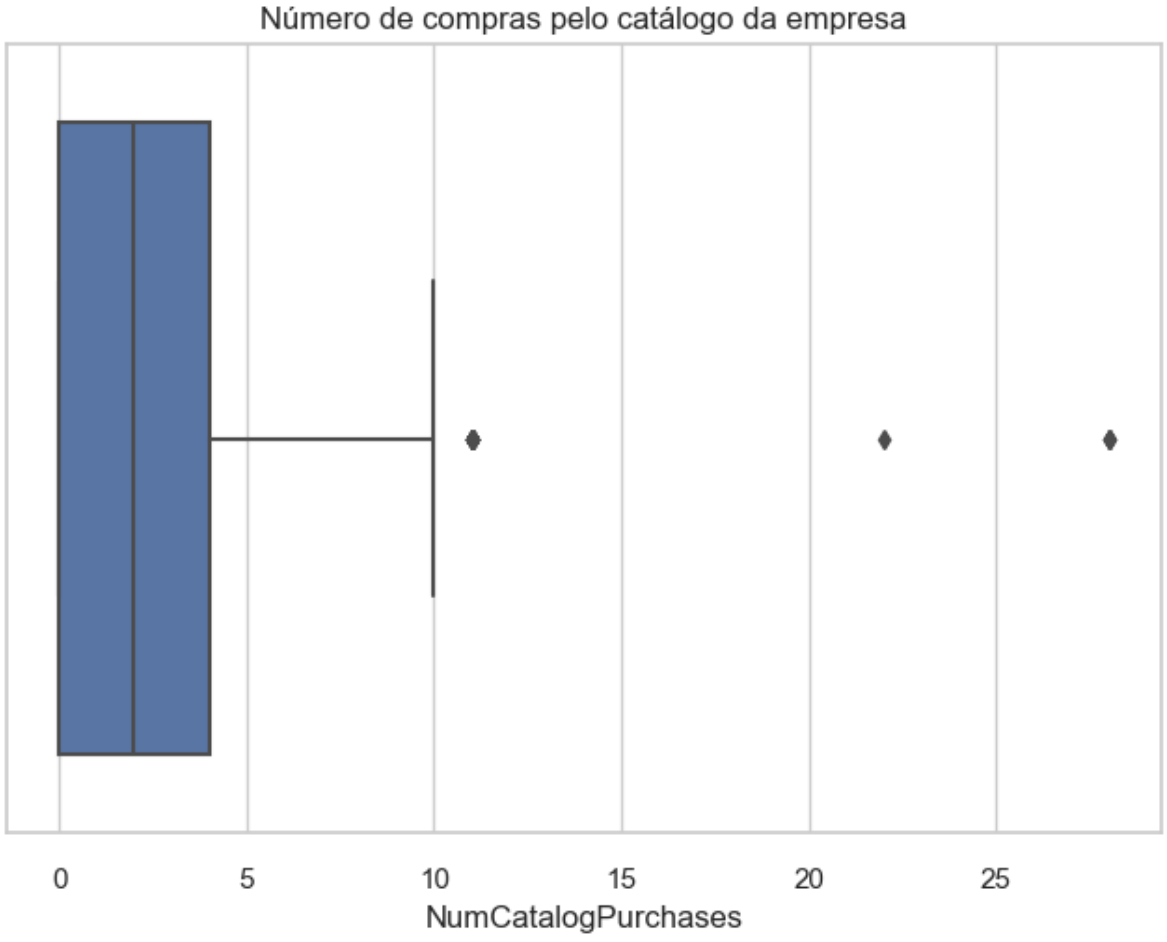


Número de compras pelo catálogo da empresa

In [ ]:
```
sns.displot(data=data, x = 'NumCatalogPurchases', color='#59A590', kde=True).set(t
```

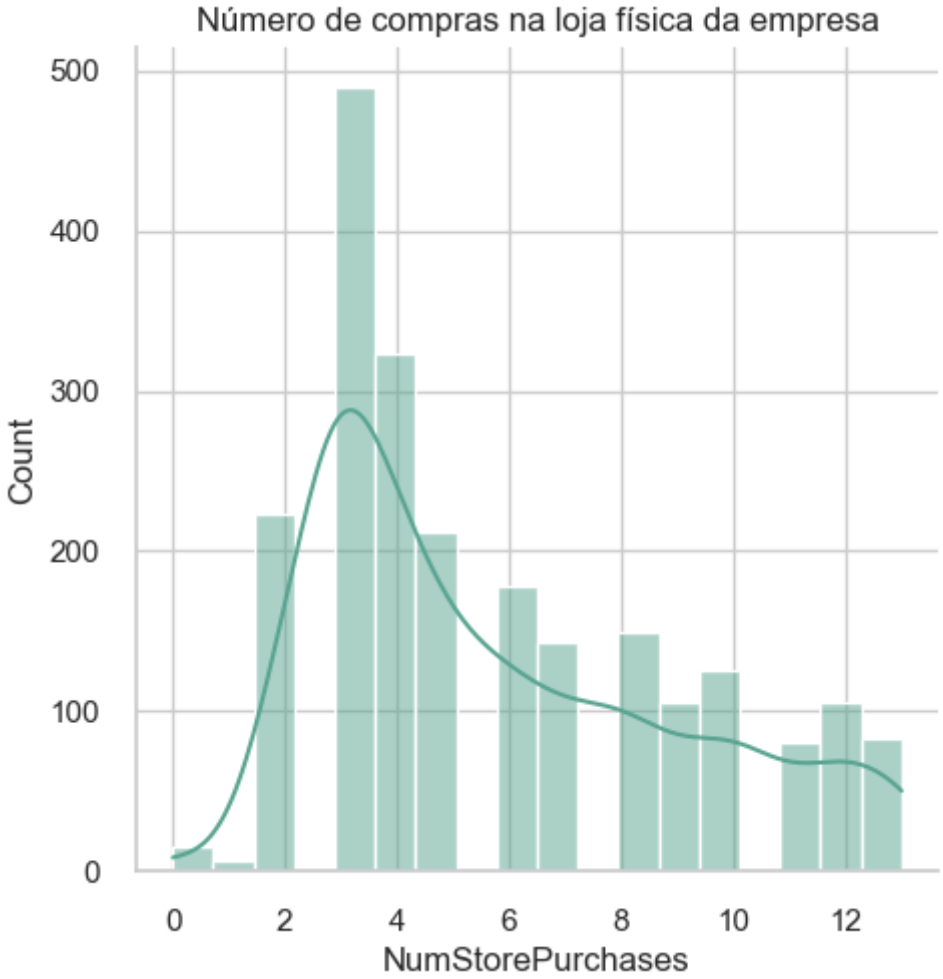


In [ ]:
```
sns.boxplot(x=data['NumCatalogPurchases']).set(title=f'Número de compras pelo catál
```

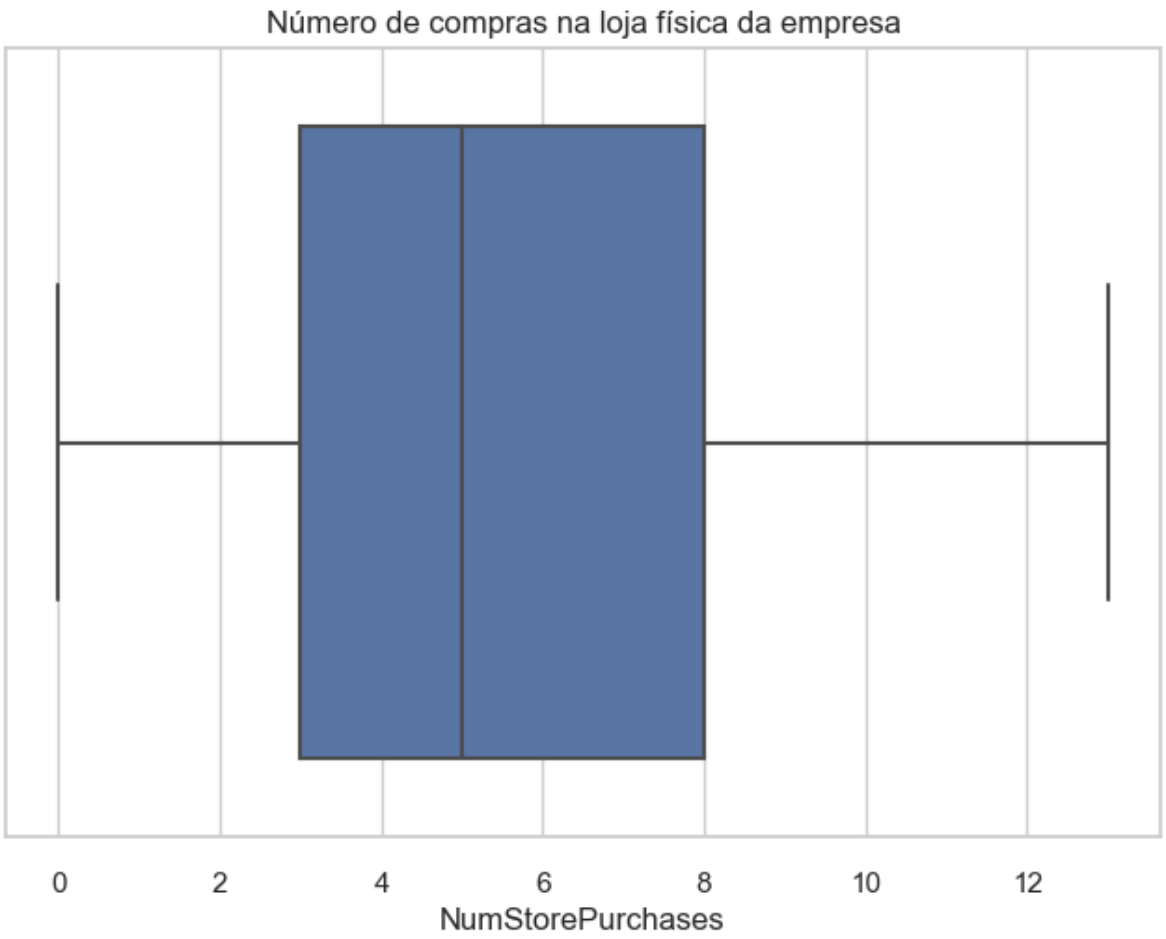


Número de compras na loja física da empresa

In [ ]:
```
sns.displot(data=data, x = 'NumStorePurchases', color='#59A590', kde=True).set(titl
```



In [ ]:
```
sns.boxplot(x=data['NumStorePurchases']).set(title=f'Número de compras na loja físi
```



Taxa de conversão dos clientes conforme oferta do marketing

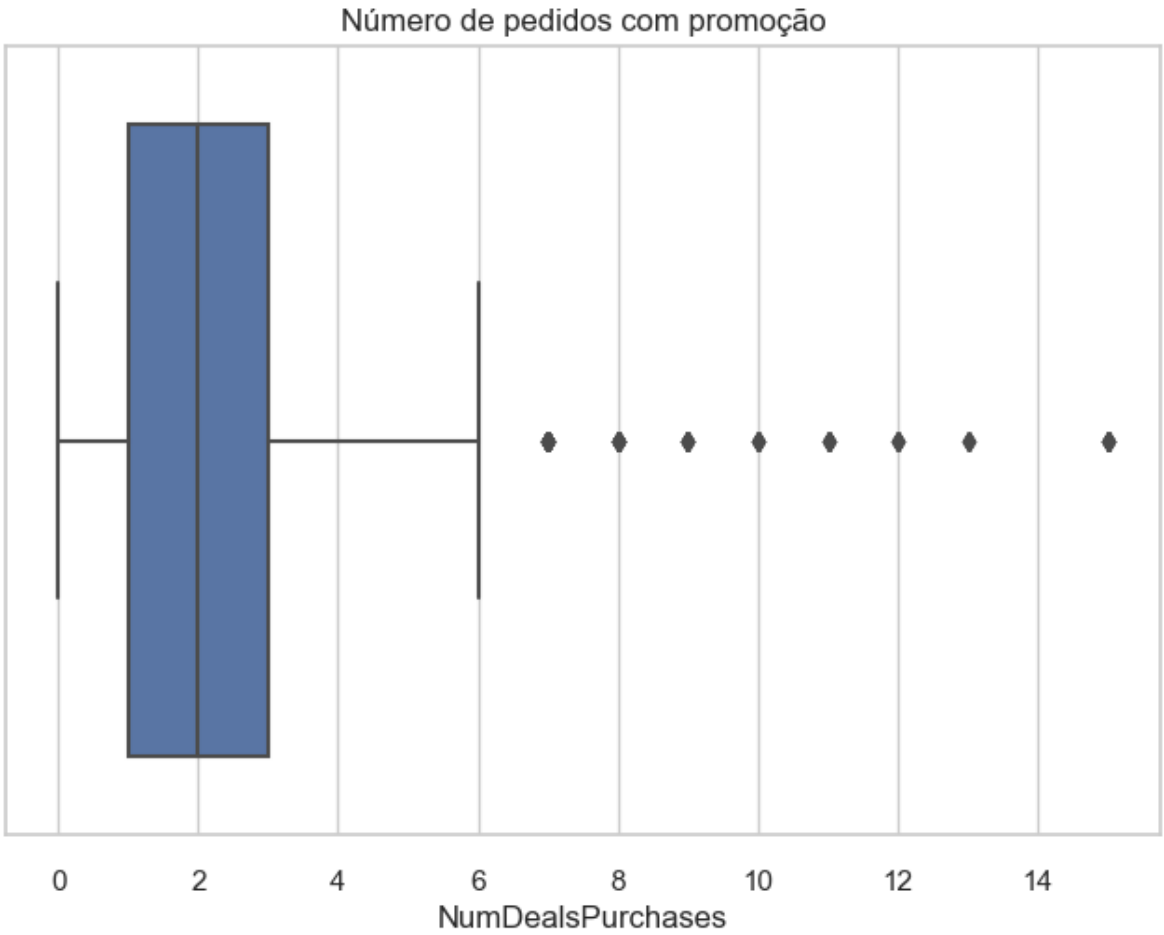
In [ ]:
```
clients = data.shape[0]
clients_01 = data['AcceptedCmp1'].value_counts()[1]
clients_02 = data['AcceptedCmp2'].value_counts()[1]
clients_03 = data['AcceptedCmp3'].value_counts()[1]
clients_04 = data['AcceptedCmp4'].value_counts()[1]
clients_05 = data['AcceptedCmp5'].value_counts()[1]
clients_06 = data['Response'].value_counts()[1]
```

In [ ]:
```
print(f'No funil de vendas das campanhas de marketing, estas são as seguintes taxas
print(f'Para primeira campanha {((clients_01 / clients) * 100):.2f} % aceitam a ofe
print(f'Para segunda campanha {((clients_02 / clients) * 100):.2f} % aceitam a ofer
print(f'Para terceira campanha {((clients_03 / clients) * 100):.2f} % aceitam a ofe
print(f'Para quarta campanha {((clients_04 / clients) * 100):.2f} % aceitam a ofert
print(f'Para quinta campanha {((clients_05 / clients) * 100):.2f} % aceitam a ofert
print(f'Para última campanha {((clients_06 / clients) * 100):.2f} % aceitam a ofert
```

```
No funil de vendas das campanhas de marketing, estas são as seguintes taxas de con
versão:
Para primeira campanha 6.43 % aceitam a oferta.
Para segunda campanha 1.34 % aceitam a oferta.
Para terceira campanha 7.28 % aceitam a oferta.
Para quarta campanha 7.46 % aceitam a oferta.
Para quinta campanha 7.28 % aceitam a oferta.
Para última campanha 14.91 % aceitam a oferta.
```

Número de pedidos com promoção

In [ ]:
```
sns.boxplot(x=data['NumDealsPurchases']).set(title=f'Número de pedidos com promoçã
```



Nesta variável é possível perceber que, em geral, os clientes aguardam por promoção para realizarem suas compras.

Etapa 04 - Realize o pré-processamento adequado dos dados. Descreva os passos necessários.

1. Inicialmente são observadas as features através das suas estatísticas
2. Os valores nulos são removidos
3. É realizada uma breve feature engineering para obtenção da idade do cliente e do número de filhos
4. Após isso, são removidas as features desnecessárias

Exploração dos dados

In [ ]:
```
data.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	
ID	2240.0	5592.159821	3246.662198	0.0	2828.25	5458.5	8427.75	1
Year_Birth	2240.0	1968.805804	11.984069	1893.0	1959.00	1970.0	1977.00	
Income	2216.0	52247.251354	25173.076661	1730.0	35303.00	51381.5	68522.00	66
Kidhome	2240.0	0.444196	0.538398	0.0	0.00	0.0	1.00	
Teenhome	2240.0	0.506250	0.544538	0.0	0.00	0.0	1.00	
Recency	2240.0	49.109375	28.962453	0.0	24.00	49.0	74.00	
MntWines	2240.0	303.935714	336.597393	0.0	23.75	173.5	504.25	
MntFruits	2240.0	26.302232	39.773434	0.0	1.00	8.0	33.00	
MntMeatProducts	2240.0	166.950000	225.715373	0.0	16.00	67.0	232.00	
MntFishProducts	2240.0	37.525446	54.628979	0.0	3.00	12.0	50.00	
MntSweetProducts	2240.0	27.062946	41.280498	0.0	1.00	8.0	33.00	
MntGoldProds	2240.0	44.021875	52.167439	0.0	9.00	24.0	56.00	
NumDealsPurchases	2240.0	2.325000	1.932238	0.0	1.00	2.0	3.00	
NumWebPurchases	2240.0	4.084821	2.778714	0.0	2.00	4.0	6.00	
NumCatalogPurchases	2240.0	2.662054	2.923101	0.0	0.00	2.0	4.00	
NumStorePurchases	2240.0	5.790179	3.250958	0.0	3.00	5.0	8.00	
NumWebVisitsMonth	2240.0	5.316518	2.426645	0.0	3.00	6.0	7.00	
AcceptedCmp3	2240.0	0.072768	0.259813	0.0	0.00	0.0	0.00	
AcceptedCmp4	2240.0	0.074554	0.262728	0.0	0.00	0.0	0.00	
AcceptedCmp5	2240.0	0.072768	0.259813	0.0	0.00	0.0	0.00	
AcceptedCmp1	2240.0	0.064286	0.245316	0.0	0.00	0.0	0.00	
AcceptedCmp2	2240.0	0.013393	0.114976	0.0	0.00	0.0	0.00	
Complain	2240.0	0.009375	0.096391	0.0	0.00	0.0	0.00	
Z_CostContact	2240.0	3.000000	0.000000	3.0	3.00	3.0	3.00	
Z_Revenue	2240.0	11.000000	0.000000	11.0	11.00	11.0	11.00	
Response	2240.0	0.149107	0.356274	0.0	0.00	0.0	0.00	

Remoção de nulos

In [ ]:
```
data = data.dropna()
print(f'O novo número total de clientes únicos observados é de {data.shape[0]} clie
```

```
O novo número total de clientes únicos observados é de 2216 clientes.
```

Feature Engineering

In [ ]:
```
# Criação da variável idade
date = datetime.date.today()
```

```
data['Age'] = date.year - data['Year_Birth']
data.head()
```

Out[ ]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	

5 rows × 30 columns

In [ ]:
```
# Identificar a distribuição do grau de escolaridade
data['Education'].value_counts()
```

Out[ ]:
```
Graduation    1116
PhD            481
Master         365
2n Cycle       200
Basic           54
Name: Education, dtype: int64
```

In [ ]:
```
# Identificar a distribuição do estado civil
data['Marital_Status'].value_counts()
```

Out[ ]:
```
Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

In [ ]:
```
# Condensar o número de filhos em uma variável
data['Children'] = data['Kidhome'] + data['Teenhome']
data.head()
```

Out[ ]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rece
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	

5 rows × 31 columns

Remoção de outliers

In [ ]:
```
data.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	
ID	2216.0	5588.353339	3249.376275	0.0	2814.75	5458.5	8421.75	1
Year_Birth	2216.0	1968.820397	11.985554	1893.0	1959.00	1970.0	1977.00	
Income	2216.0	52247.251354	25173.076661	1730.0	35303.00	51381.5	68522.00	66
Kidhome	2216.0	0.441787	0.536896	0.0	0.00	0.0	1.00	
Teenhome	2216.0	0.505415	0.544181	0.0	0.00	0.0	1.00	
Recency	2216.0	49.012635	28.948352	0.0	24.00	49.0	74.00	
MntWines	2216.0	305.091606	337.327920	0.0	24.00	174.5	505.00	
MntFruits	2216.0	26.356047	39.793917	0.0	2.00	8.0	33.00	
MntMeatProducts	2216.0	166.995939	224.283273	0.0	16.00	68.0	232.25	
MntFishProducts	2216.0	37.637635	54.752082	0.0	3.00	12.0	50.00	
MntSweetProducts	2216.0	27.028881	41.072046	0.0	1.00	8.0	33.00	
MntGoldProds	2216.0	43.965253	51.815414	0.0	9.00	24.5	56.00	
NumDealsPurchases	2216.0	2.323556	1.923716	0.0	1.00	2.0	3.00	
NumWebPurchases	2216.0	4.085289	2.740951	0.0	2.00	4.0	6.00	
NumCatalogPurchases	2216.0	2.671029	2.926734	0.0	0.00	2.0	4.00	
NumStorePurchases	2216.0	5.800993	3.250785	0.0	3.00	5.0	8.00	
NumWebVisitsMonth	2216.0	5.319043	2.425359	0.0	3.00	6.0	7.00	
AcceptedCmp3	2216.0	0.073556	0.261106	0.0	0.00	0.0	0.00	
AcceptedCmp4	2216.0	0.074007	0.261842	0.0	0.00	0.0	0.00	
AcceptedCmp5	2216.0	0.073105	0.260367	0.0	0.00	0.0	0.00	
AcceptedCmp1	2216.0	0.064079	0.244950	0.0	0.00	0.0	0.00	
AcceptedCmp2	2216.0	0.013538	0.115588	0.0	0.00	0.0	0.00	
Complain	2216.0	0.009477	0.096907	0.0	0.00	0.0	0.00	
Z_CostContact	2216.0	3.000000	0.000000	3.0	3.00	3.0	3.00	
Z_Revenue	2216.0	11.000000	0.000000	11.0	11.00	11.0	11.00	
Response	2216.0	0.150271	0.357417	0.0	0.00	0.0	0.00	
Age	2216.0	54.179603	11.985554	27.0	46.00	53.0	64.00	
Children	2216.0	0.947202	0.749062	0.0	0.00	1.0	1.00	

Na descrição estatística dos dados, é possível notar alguns outliers:

- O cliente de maior idade possui 130 anos
- O cliente de maior renda possui a renda muito superior ao quartil 75

Além disso, deve-se remover as features: Z_CostContact e Z_Revenue

In [ ]:
```
# Remoção dos Outliers na Feature "Age" e "Income"
data = data[(data['Age'] < 90)]
data = data[(data['Income'] < 600000)]
print(f'O novo número total de clientes únicos observados é de {data.shape[0]} clie
```

```
O novo número total de clientes únicos observados é de 2212 clientes.
```

Remoção de features

In [ ]:
```
data.columns
```

Out[ ]:
```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response',
       'Age', 'Children'],
      dtype='object')
```

In [ ]:
```
# Remoção das features desnecessárias
data = data.drop(['Kidhome', 'Teenhome', 'Year_Birth', 'Z_CostContact', 'Z_Revenue
```

Seleção das features para clusterização

In [ ]:
```
data.head()
```

Out[ ]:

	Education	Marital_Status	Income	Dt_Customer	Recency	MntWines	MntFruits	MntMeatPro
0	Graduation	Single	58138.0	04-09-2012	58	635	88	
1	Graduation	Single	46344.0	08-03-2014	38	11	1	
2	Graduation	Together	71613.0	21-08-2013	26	426	49	
3	Graduation	Together	26646.0	10-02-2014	26	11	4	
4	PhD	Married	58293.0	19-01-2014	94	173	43	

5 rows × 25 columns

In [ ]:
```
data.columns
```

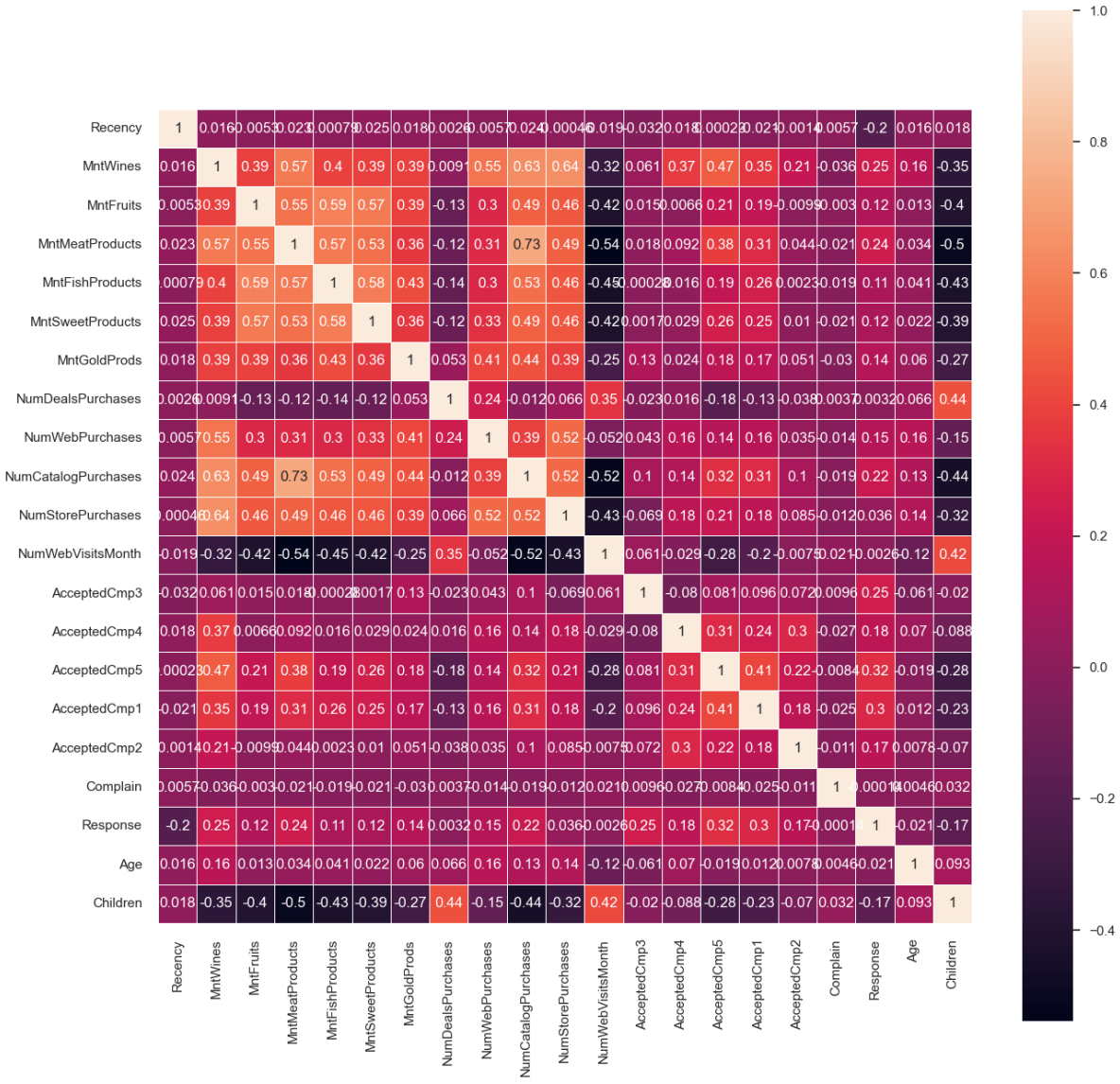
Out[ ]:
```
Index(['Education', 'Marital_Status', 'Income', 'Dt_Customer', 'Recency',
       'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',
       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
       'NumWebVisitsMonth', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
       'AcceptedCmp1', 'AcceptedCmp2', 'Complain', 'Response', 'Age',
       'Children'],
      dtype='object')
```

In [ ]:
```
df = data[['Recency','MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
```

Normalização dos dados

In [ ]:
```
# 01 - Normalização dos dados
normalized_data = (df - df.mean()) / df.std()

X = normalized_data.values
```

In [ ]:
```
corr = normalized_data.corr()
f, ax = plt.subplots(figsize=(15, 15))
sns.heatmap(corr, square=True, linewidths=.5, annot=True);
```



A quantidade de dias da última compra e a quantidade de reclamações não possui correlação expressiva com as demais variáveis.
Os valores gastos, exceto com produtos premium, se relacionam minimamente entre si.
O gasto com carnes ou alimentos para preparo de refeições está fortemente associado à compra por catálogos.
A variável filhos possui certa correlação com compras com descontos e número de visitas ao site da empresa por mês.

Etapa 05 - Clusterização

1. Realizar o agrupamento dos dados, escolhendo o número ótimo de clusters. Para tal, use o índice de silhueta e as técnicas:

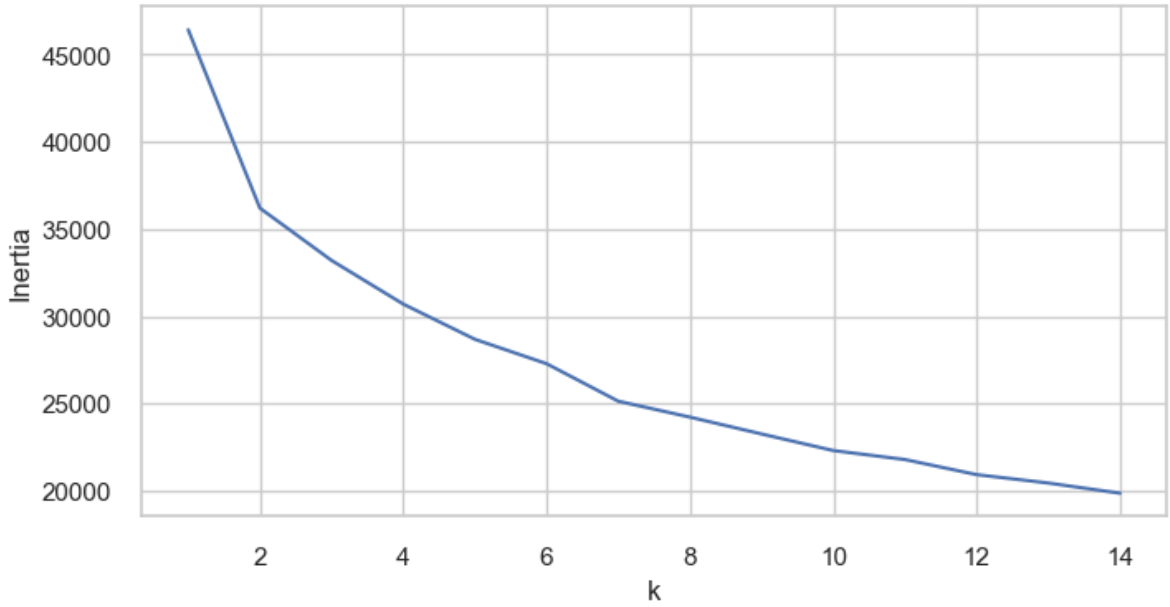
- K-Médias
- DBScan

KMEANS

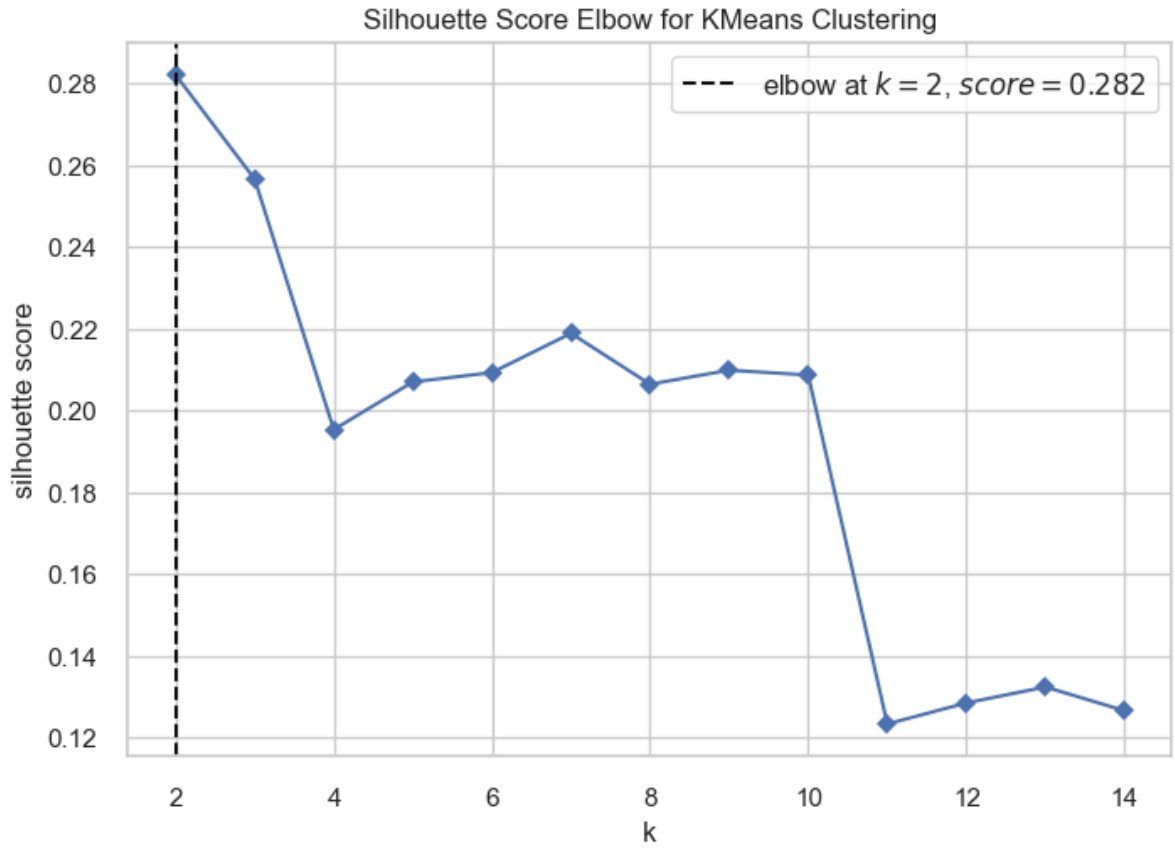
In [ ]:
```
seed = 42
k_min = 1
k_max = 15
```

In [ ]:
```
def plot_inertia(df, kmin=k_min, kmax=k_max, figsize=(8, 4)):

    _range = range(kmin, kmax)
    inertias = []
    for k in _range:
        kmeans = KMeans(n_init = 10, n_clusters=k, random_state=seed)
        kmeans.fit(df)
        inertias.append(kmeans.inertia_)

    plt.figure(figsize=figsize)
    plt.plot(_range, inertias, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Inertia')
    plt.show()
```

In [ ]:
```
plot_inertia(X);
```

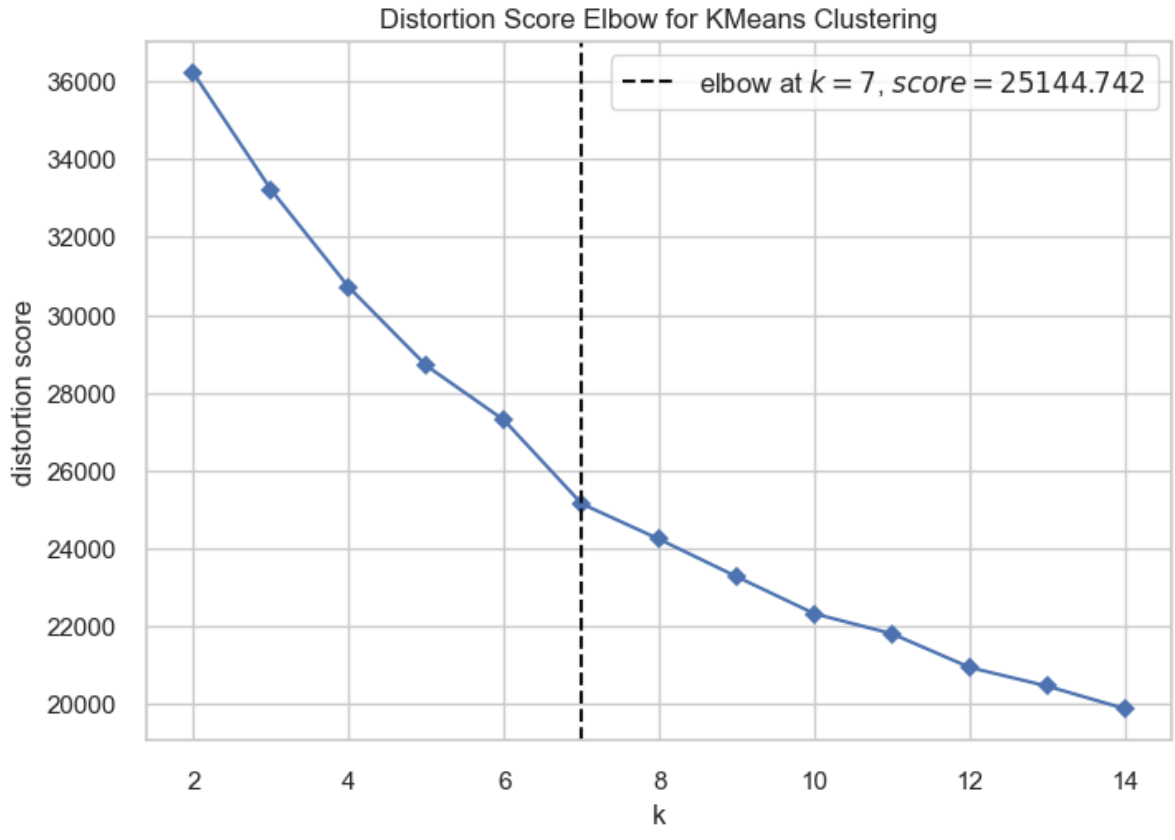


In [ ]:
```
from yellowbrick.cluster import KElbowVisualizer

kmeans = KMeans(n_init = 10, random_state=seed)
visualizer = KElbowVisualizer(
    kmeans, k=(2,15), metric='silhouette', timings=False
)

visualizer.fit(X)
visualizer.show()
```



Out[ ]:
```
<AxesSubplot: title={'center': 'Silhouette Score Elbow for KMeans Clustering'}, xl
abel='k', ylabel='silhouette score'>
```

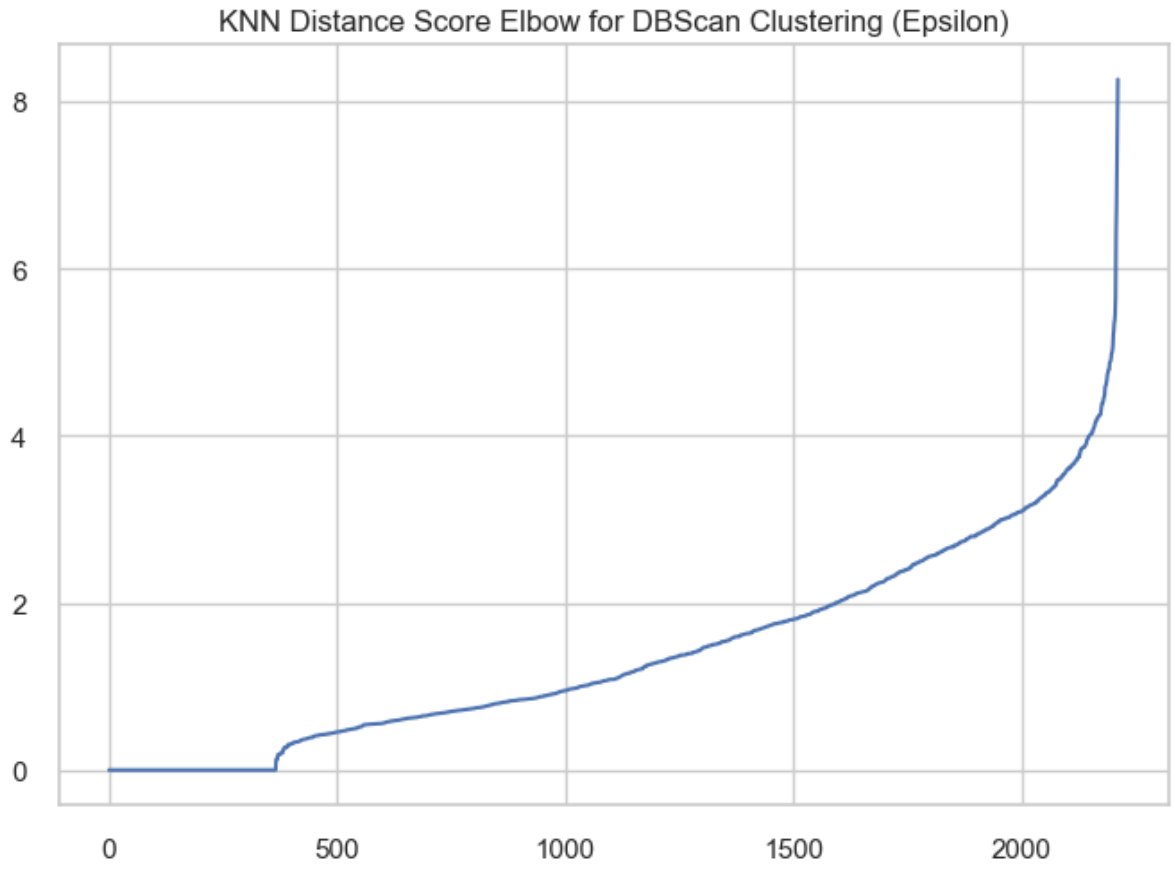
In [ ]:
```
kmeans = KMeans(n_init = 10, random_state=seed)
visualizer = KElbowVisualizer(
    kmeans, k=(2,15), metric='distortion', timings=False
)
visualizer.fit(X)
visualizer.show()
```



Out[ ]:
```
<AxesSubplot: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xl
abel='k', ylabel='distortion score'>
```

DBSCAN

In [ ]:
```
knn = NearestNeighbors(n_neighbors= 12).fit(X)
distances, indices = knn.kneighbors(X)
distances = np.sort(distances, axis=0)[:,1]
plt.title('KNN Distance Score Elbow for DBScan Clustering (Epsilon)')
plt.plot(distances);
```



In [ ]:
```
epsilon = 4.1
min_samples = 20
```

In [ ]:
```
dbscan_model = DBSCAN(eps=epsilon, min_samples= min_samples).fit(X)

no_clusters = len(np.unique(dbscan_model.labels_))
no_noise = np.sum(np.array(dbscan_model.labels_) == -1, axis=0)
```

In [ ]:
```
print(f'DBSCAN encontrou {no_clusters} clusters, com epsilon = {epsilon} e amostra
```

```
DBSCAN encontrou 4 clusters, com epsilon = 4.1 e amostra mínima = 20.
```

Com os resultados em mão, descreva o processo de mensuração do índice de silhueta. Mostre o gráfico e justifique o número de clusters escolhidos.

```
O processo de mensuração do índice de silhueta basicamente
funciona assim:
- É uma medida normalizada e varia entre -1 e 1, onde:

   *  1 : perfeição
```

* 0 : confusão
* -1 : erro ou associação errada

A silhueta não é uma boa medida de validação de clusters não convexos.

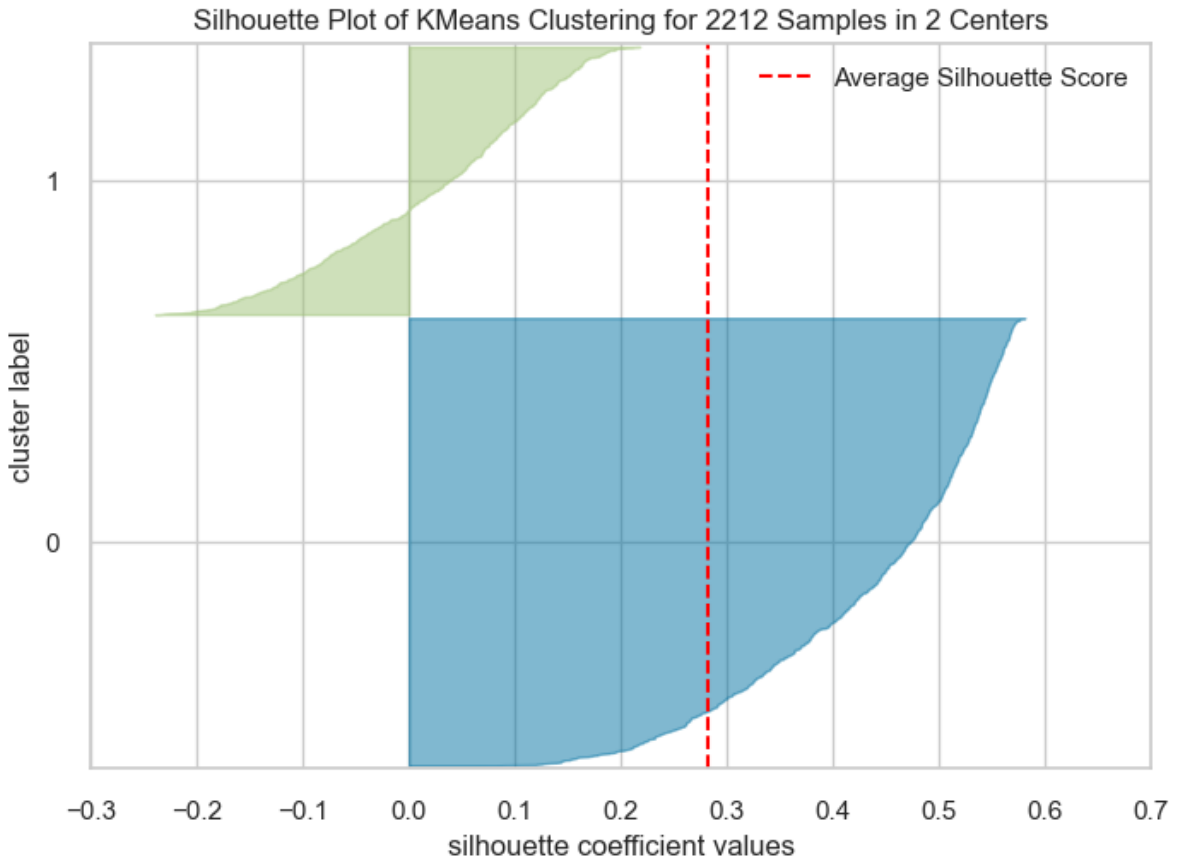
Clientes com silhueta menor ou igual a zero devem ser analisados.

Para kmeans, no primeiro gráfico do cotovelo considerando a inércia, os números de k óbvios são 2 e 7.
Os mesmos valores aparecem na visualização do índice de silhueta, mostrando k = 2, e no índice de distorção, mostrando k = 7. No entanto, uma escolha mais correta de acordo com o índice de silhueta, seriam 4 clusters.

Para o DBSCAN, observado o gráfico relacionado ao epsilon, calcula-se o resultado de epsilon = 4.1 e número de 20 amostras, devido ao tamanho da base. (Número 20 foi escolhido devido ao tamanho da base)

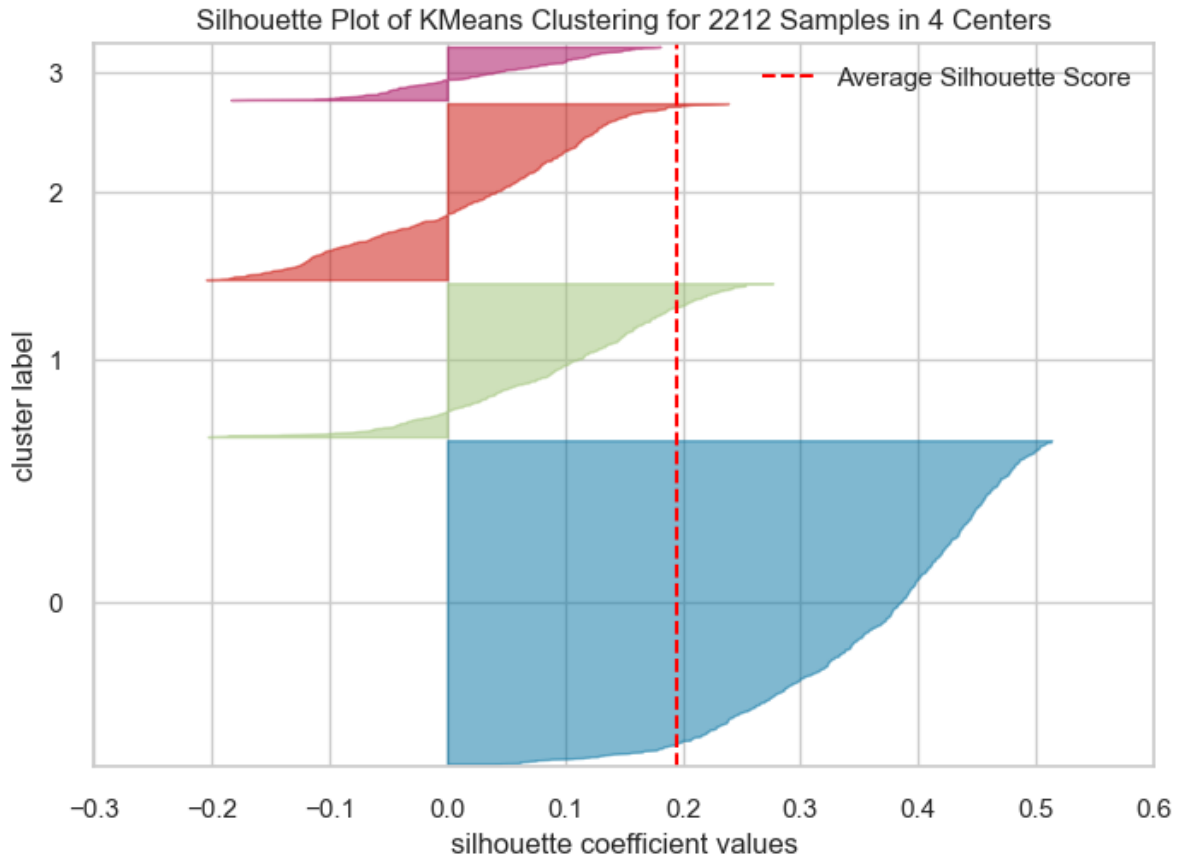
2. Compare os dois resultados, aponte as semelhanças e diferenças e interprete.

In [ ]:
```
# Plot da silhueta para 2  clusters
visualizer = SilhouetteVisualizer( KMeans(n_clusters = 2, n_init = 10, random_state
visualizer.fit(X)
visualizer.show();
```

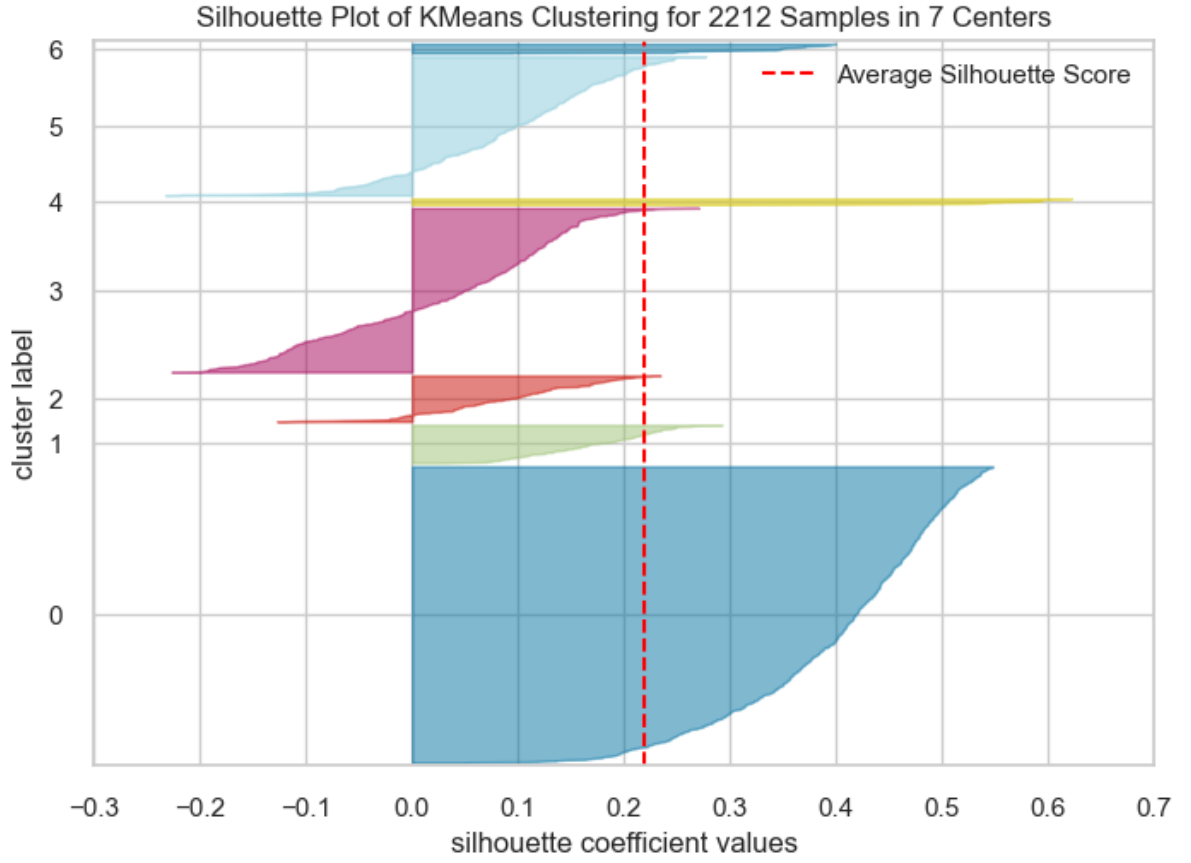


In [ ]:
```
# Plot da silhueta para 4  clusters
visualizer = SilhouetteVisualizer( KMeans(n_clusters = 4, n_init = 10, random_state
```

```
visualizer.fit(X)
visualizer.show();
```



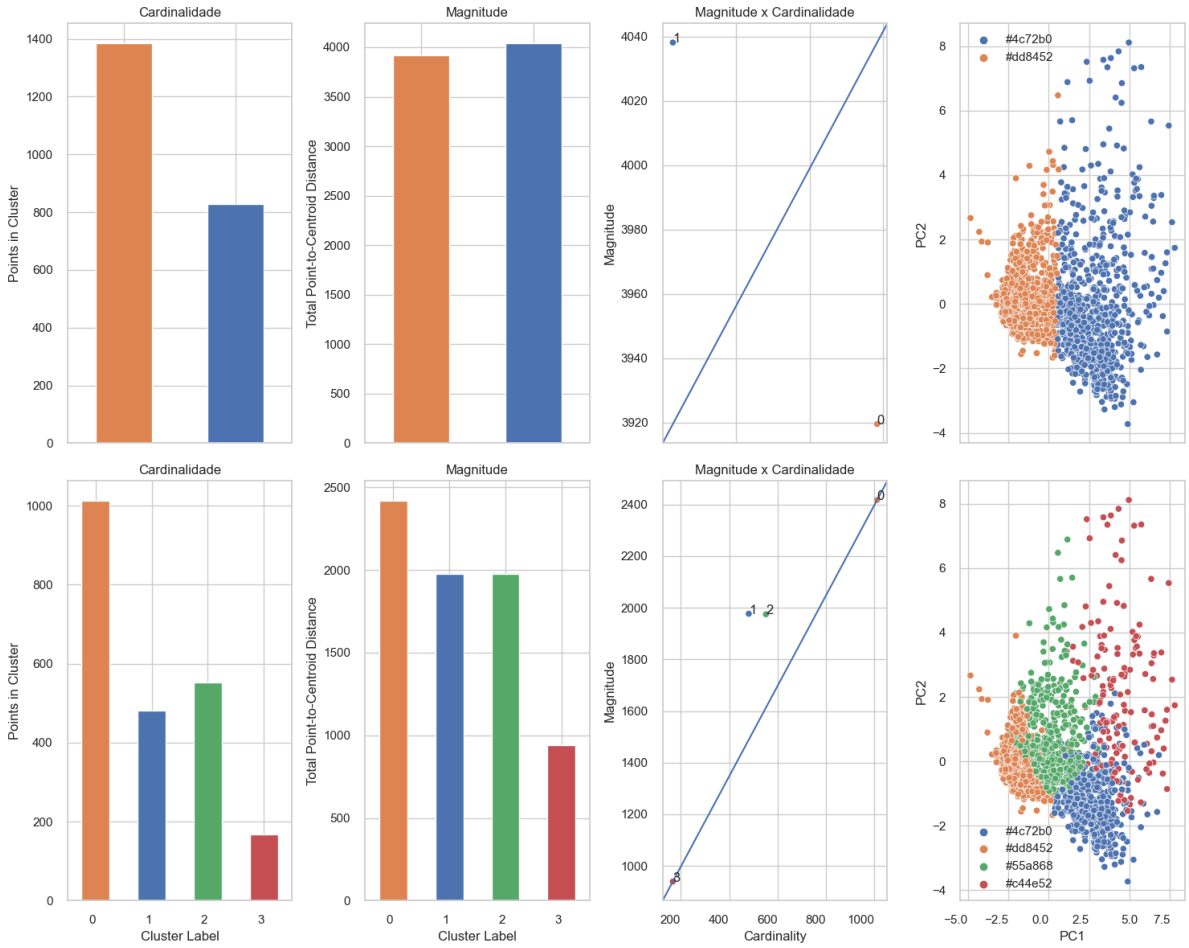
In [ ]:
```
# Plot da silhueta para 7 clusters
visualizer = SilhouetteVisualizer( KMeans(n_clusters = 7, n_init = 10, random_state
visualizer.fit(X)
visualizer.show();
```



O índice de silhueta piora conforme o aumento do número de clusters. O único cluster que se mantém parcialmente coeso é o 0. Os demais clusters apresentam muitos pontos em zona de confusão. Para exemplificar isto, os gráficos abaixo mostram algumas medidas de validação do número de clusters escolhidos, desconsiderando o k = 7.

3. Escolha mais duas medidas de validação para comparar com o índice de silhueta e analise os resultados encontrados. Observe, para a escolha, medidas adequadas aos algoritmos.

In [ ]:
```
fig, axs = plt.subplots(2, 4, figsize=(15, 12))

for row_idx, k in enumerate([2, 4]):

    row_axs = axs[row_idx]

    kmeans = KMeans(n_init = 10, n_clusters=k, random_state=seed)
    k_fit = kmeans.fit(X)

    cluster_colors = list(set(map(lambda x: '#000' if x == -1 else sns.color_palett

    plot_cluster_cardinality(k_fit.labels_,
                             ax=row_axs[0],
                             title="Cardinalidade",
                             color=cluster_colors
                            )
    plot_cluster_magnitude(X,
                           k_fit.labels_,
                           k_fit.cluster_centers_,
                           euclidean,
                           ax=row_axs[1],
                           title="Magnitude",
                           color=cluster_colors
                          )
    plot_magnitude_vs_cardinality(X,
                                  k_fit.labels_,
                                  k_fit.cluster_centers_,
                                  euclidean,
                                  color=cluster_colors[0:k_fit.n_clusters],
                                  ax=row_axs[2],
                                  title="Magnitude x Cardinalidade")

    plot_cluster_points(X, k_fit.labels_,
                        hue=list(map(lambda x: cluster_colors[x], k_fit.labels_)),


fig.autofmt_xdate(rotation=0)
plt.tight_layout()
```

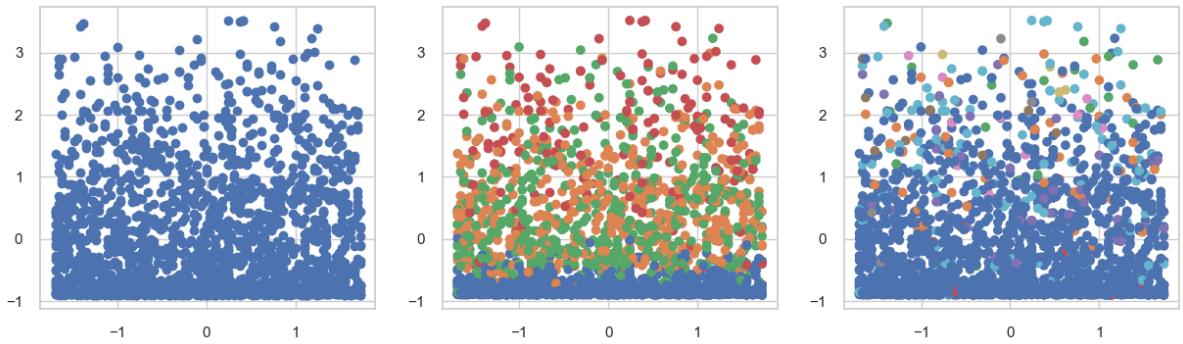


Percebe-se nos gráficos acima o problema da correlação entre cardinalidade e magnitude, em especial para k = 2. O algoritmo K-Means simplesmente separa os dados em 2 clusters de magnitude semelhante, porém quando olhamos o gráfico de PCA, é possível notar que a classificação não demonstra um bom desempenho.

Já para k = 4, o gráfico de PCA demonstra uma distribuição dos pontos entre os clusters performando melhor. No entanto, ainda existem muitos pontos de confusão. Essa conclusão complementa o gráfico de silhueta para k = 4, que possui média próxima de 0.2, o que está mais próximo de confusão do que perfeição.

In [ ]:
```
kmeans_4 = KMeans(n_clusters = 4, n_init = 10, random_state = seed)
kmeans_labels = kmeans_4.fit_predict(X)

dbscan =  DBSCAN(eps=epsilon)
dbscan_labels = dbscan.fit_predict(X)

fig, axs = plt.subplots(1, 3, figsize=(15, 4))

axs[0].scatter(X[:,0], X[:,1])
axs[1].scatter(X[:,0], X[:,1], c=[sns.color_palette().as_hex()[l] for l in kmeans_l
axs[2].scatter(X[:,0], X[:,1], c=[sns.color_palette().as_hex()[l] for l in dbscan_l

kmeans_score = DBCV(X, kmeans_labels, dist_function=euclidean)
dbscan_score = DBCV(X, dbscan_labels, dist_function=euclidean)
print(kmeans_score, dbscan_score)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[177], line 13
     10 axs[1].scatter(X[:,0], X[:,1], c=[sns.color_palette().as_hex()[l] for l in
kmeans_labels])
     11 axs[2].scatter(X[:,0], X[:,1], c=[sns.color_palette().as_hex()[l] for l in
dbscan_labels])
---> 13 kmeans_score = DBCV(X, kmeans_labels, dist_function=euclidean)
     14 dbscan_score = DBCV(X, dbscan_labels, dist_function=euclidean)
     15 print(kmeans_score, dbscan_score)

File c:\Users\natha\Desktop\PF Validação de modelos de clustering\validacao-modelo
s-clusterizacao\env\lib\site-packages\DBCV\DBCV.py:30, in DBCV(X, labels, dist_fun
ction)
     16 def DBCV(X, labels, dist_function=euclidean):
     17     """
     18     Density Based clustering validation
     19
   (...)
     28         score in range[-1, 1] indicating validity of clustering assignment
s
     29     """
---> 30     graph = _mutual_reach_dist_graph(X, labels, dist_function)
     31     mst = _mutual_reach_dist_MST(graph)
     32     cluster_validity = _clustering_validity_index(mst, labels)

File c:\Users\natha\Desktop\PF Validação de modelos de clustering\validacao-modelo
s-clusterizacao\env\lib\site-packages\DBCV\DBCV.py:117, in _mutual_reach_dist_grap
h(X, labels, dist_function)
    115 class_i = labels[row]
    116 class_j = labels[col]
--> 117 members_i = _get_label_members(X, labels, class_i)
    118 members_j = _get_label_members(X, labels, class_j)
    119 dist = _mutual_reachability_dist(point_i, point_j,
    120                                  members_i, members_j,
    121                                  dist_function)

File c:\Users\natha\Desktop\PF Validação de modelos de clustering\validacao-modelo
s-clusterizacao\env\lib\site-packages\DBCV\DBCV.py:258, in _get_label_members(X, l
abels, cluster)
    244 """
    245 Helper function to get samples of a specified cluster.
    246
   (...)
    255     specified cluster.
    256 """
    257 indices = np.where(labels == cluster)[0]
--> 258 members = X[indices]
    259 return members

KeyboardInterrupt:
```



A métrica de avaliação de qualidade baseado em densidade (DBCV) não performou bem para a base de dados. Ao interromper a execução é possível visualizar parcialmente os pontos distribuídos no primeiro gráfico, no segundo a clusterização por k-means e no terceiro a clusterização pelo DBSCAN.

Para esta base de dados, eu realizaria uma clusterização com k = 2, separaria a base entre este dois clusters e testaria novas configurações de clusterização entre as novas bases criadas, em especial para o cluster 0, que teve uma coesão maior na análise do índice de silhueta, cardinalidade e magnitude.

Não acredito que a base esteja preparada para realização da clusterização, sendo necessário, na minha opinião, realizar a estratégia descrita acima e talvez trabalhar melhor as features.

4. Realizando a análise, responda: A silhueta é um o índice indicado para escolher o número de clusters para o algoritmo de DBScan?

Não, pois ela penaliza os pontos da base de dados baseado na posição dos centróides e é fortemente indicada para bases de dados cujos clusters são convexos, que são alvos do K-Means

Etapa 04 - Medidas de similaridade

1. Um determinado problema, apresenta 10 séries temporais distintas. Gostaríamos de agrupá-las em 3 grupos, de acordo com um critério de similaridade, baseado no valor máximo de correlação cruzada entre elas. Descreva em tópicos todos os passos necessários.

- Explorar a base de dados para entender o comportamento dos dados
- Determinar alguns critérios para exploração da correlação entre as séries, como a janela de atuação dessa correlação cruzada
- Obter a correlação entre as 10 séries, escolhendo o valor máximo da correlação quando comparadas às demais séries
- Inserir os valores de correlação máxima em uma matriz, semelhante ao processo da camada de pooling das CNNs
- Repetir o processo até a obtenção de todas as correlações

máximas entre as séries temporais
- A partir desta matriz de máximas correlações, aplicaria talvez K-Means para identificar os 3 clusters desejados

2. Para o problema da questão anterior, indique qual algoritmo de clusterização você usaria. Justifique.

Windowed Time Lagged Cross Correlation, junto com K-Means.

3. Indique um caso de uso para essa solução projetada.

Conforme este artigo (https://towardsdatascience.com/four-ways-to-quantify-synchrony-between-time-series-data-b99136c4a9c9), são analisados os sinais neurais de duas pessoas conversando. A ideia se aplicaria a correlacionar um grupo de amigos e a dinâmica entre suas interações. Não tenho certeza do desempenho desse estudo, mas é uma possibilidade.

4. Sugira outra estratégia para medir a similaridade entre séries temporais. Descreva em tópicos os passos necessários.

Cálculo da corelação de Pearson entre duas séries temporais.
- Selecionar duas séries temporais
- Calcular a correlação entre elas
- Obtém-se a correlação geral de Pearson entre as séries

Também é possível incluir o janelamento entre as séries e a correlação nesta faixa de pontos selecionados.