```
from google.colab import drive

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings("ignore")



from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, ConfusionMatrixDisplay


# Configurações de estilo
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (8,5)


# Criar um DataFrame a partir de um arquivo CSV
df = pd.read_csv('/content/dados_processados_final.csv', sep=',')
print(df.head())


# Print column names to verify
print(df.columns)
```

```
   id_cliente  idade       sexo       escolaridade   profissao  \
0        1001     34   Feminino  Superior Completo   professor
1        1002     28  Masculino              Médio    vendedor
2        1003     45   Feminino      Pós-graduação      médico
3        1004     52  Masculino  Superior Completo  engenheiro
4        1005     29   Feminino              Médio  secretária

   autoavaliacao_saude grupo_caminhada  gasto_produtos_naturais  \
0                  4.0             Sim                    250.5
1                  3.0             Não                    180.2
2                  5.0             não                    420.8
3                  3.0             Sim                    310.4
4                  4.0             Não                    190.6

   gasto_produtos_ultraprocessados  frequencia_compra  ...  \
0                           130.75                  5  ...
1                           220.30                  8  ...
2                            80.15                  3  ...
3                           160.90                  4  ...
4                           185.50                  6  ...

  faixa_etaria_meia_idade faixa_etaria_idoso  idade_zscore  perfil_saudavel  \
0                   False              False     -0.504597                0
1                   False              False     -1.206109                0
2                    True              False      0.781510                0
3                    True              False      1.599941                0
4                   False              False     -1.089191                0

   faixa_etaria  alto_consumo_natural  faixa_etaria_otimizada  cluster  \
0        adulto                     0    Faixa 2 (30-38 anos)        2
1        adulto                     0     Faixa 1 (0-29 anos)        0
2   meia_idade                      1    Faixa 3 (39-46 anos)        1
3   meia_idade                      1    Faixa 4 (47-57 anos)        2
4        adulto                     0     Faixa 1 (0-29 anos)        2

   perfil_consumidor consumidor_natural_class
0              Misto                        1
1    Industrializado                        0
2           Saudável                        1
3              Misto                        1
4              Misto                        1

[5 rows x 25 columns]
Index(['id_cliente', 'idade', 'sexo', 'escolaridade', 'profissao',
       'autoavaliacao_saude', 'grupo_caminhada', 'gasto_produtos_naturais',
       'gasto_produtos_ultraprocessados', 'frequencia_compra',
       'participa_programa_saude', 'regiao', 'participa_caminhada',
       'profissao_encoded', 'faixa_etaria_adulto', 'faixa_etaria_meia_idade',
       'faixa_etaria_idoso', 'idade_zscore', 'perfil_saudavel', 'faixa_etaria',
       'alto_consumo_natural', 'faixa_etaria_otimizada', 'cluster',
       'perfil_consumidor', 'consumidor_natural_class'],
```

```
           dtype='object')
```

```
# 2. Pré-processamento de Dados

# Verificar valores ausentes
print(df.isnull().sum())

# Preencher valores ausentes
df['grupo_caminhada'] = df['grupo_caminhada'].fillna('não')
df['profissao'] = df['profissao'].fillna('desconhecida')
df['autoavaliacao_saude'] = df['autoavaliacao_saude'].fillna(df['autoavaliacao_saude'].mode()[0])


# Remover duplicatas
df.drop_duplicates(inplace=True)

#Padronizar texto da coluna Profissão
df['profissao'] = df['profissao'].str.lower().str.strip()

#Padronizar escolaridade
escolaridade_map= {
    'Ensino Fundamental': 'Fundamental',
    'Ensino Médio': 'Médio',
    'Ensino Superior': 'Superior',
    'Ensino Técnico': 'Técnico',
    'Pós-Graduação': 'Pós'
}

df['escolaridade'] = df['escolaridade'].replace(escolaridade_map)

print(df['escolaridade'].value_counts())
```

```
id_cliente                        0
idade                             0
sexo                              0
escolaridade                      0
profissao                         0
autoavaliacao_saude               0
grupo_caminhada                   0
gasto_produtos_naturais           0
gasto_produtos_ultraprocessados   0
frequencia_compra                 0
participa_programa_saude          0
regiao                            0
participa_caminhada               0
profissao_encoded                 0
faixa_etaria_adulto               0
faixa_etaria_meia_idade           0
faixa_etaria_idoso                0
idade_zscore                      0
perfil_saudavel                   0
faixa_etaria                      0
alto_consumo_natural              0
faixa_etaria_otimizada            0
cluster                           0
perfil_consumidor                 0
consumidor_natural_class          0
dtype: int64
escolaridade
Superior Completo    76
Médio                39
Pós-graduação        38
Fundamental          37
Name: count, dtype: int64
```

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

	0
id_cliente	0
idade	0
sexo	0
escolaridade	0
profissao	0
autoavaliacao_saude	0
grupo_caminhada	0
gasto_produtos_naturais	0
gasto_produtos_ultraprocessados	0
frequencia_compra	0
participa_programa_saude	0
regiao	0
participa_caminhada	0
profissao_encoded	0
faixa_etaria_adulto	0
faixa_etaria_meia_idade	0
faixa_etaria_idoso	0
idade_zscore	0
perfil_saudavel	0
faixa_etaria	0
alto_consumo_natural	0
faixa_etaria_otimizada	0
cluster	0
perfil_consumidor	0
consumidor_natural_class	0

dtype: int64

```
df.replace("", np.nan)
```

	id_cliente	idade	sexo	escolaridade	profissao	autoavaliacao_saude	grupo_caminhada	gasto_produtos_naturais	gasto_
0	1001	34	Feminino	Superior Completo	professor	4.0	Sim	250.5	
1	1002	28	Masculino	Médio	vendedor	3.0	Não	180.2	
2	1003	45	Feminino	Pós-graduação	médico	5.0	não	420.8	
3	1004	52	Masculino	Superior Completo	engenheiro	3.0	Sim	310.4	
4	1005	29	Feminino	Médio	secretária	4.0	Não	190.6	
...	...	...	...	...	...	...	...	...	
185	1186	35	Masculino	Superior Completo	genealogista	4.0	Sim	315.7	
186	1187	43	Feminino	Fundamental	trituradora	2.0	Não	132.9	
187	1188	34	Masculino	Pós-graduação	anestesiologista	5.0	Sim	440.9	
188	1189	28	Feminino	Superior Completo	sonoplasta	4.0	Sim	302.8	
189	1190	53	Masculino	Médio	mandrilhador	3.0	Não	168.9	

190 rows × 25 columns

```
# Substituindo valores em branco ("") e NaN por 0
df = df.replace("", 0)   # substitui apenas string vazia
df = df.fillna(0)        # substitui NaN por 0
```

```
print("\nDepois da substituição:")
print(df)
```

```
186                 2.0            Não                   132.9
187                 5.0            Sim                   440.9
188                 4.0            Sim                   302.8
189                 3.0            Não                   168.9

     gasto_produtos_ultraprocessados  frequencia_compra  ...  \
0                             130.75                  5  ...
1                             220.30                  8  ...
2                              80.15                  3  ...
3                             160.90                  4  ...
4                             185.50                  6  ...
..                               ...                ...  ...
185                           235.60                  5  ...
186                           292.80                 11  ...
187                            74.80                  2  ...
188                           220.90                  5  ...
189                           283.80                  8  ...

    faixa_etaria_meia_idade faixa_etaria_idoso  idade_zscore  perfil_saudavel  \
0                     False              False     -0.504597                0
1                     False              False     -1.206109                0
2                      True              False      0.781510                0
3                      True              False      1.599941                0
4                     False              False     -1.089191                0
..                      ...                ...           ...              ...
185                    True              False     -0.387678                0
186                    True              False      0.547672                0
187                   False              False     -0.504597                1
188                   False              False     -1.206109                0
189                    True              False      1.716860                0

     faixa_etaria  alto_consumo_natural  faixa_etaria_otimizada  cluster  \
0          adulto                     0    Faixa 2 (30-38 anos)        2
1          adulto                     0     Faixa 1 (0-29 anos)        0
2      meia_idade                     1    Faixa 3 (39-46 anos)        1
3      meia_idade                     1    Faixa 4 (47-57 anos)        2
4          adulto                     0     Faixa 1 (0-29 anos)        2
..            ...                   ...                     ...      ...
185    meia_idade                     1    Faixa 2 (30-38 anos)        2
186    meia_idade                     0    Faixa 3 (39-46 anos)        0
187        adulto                     1    Faixa 2 (30-38 anos)        1
188        adulto                     1     Faixa 1 (0-29 anos)        2
189    meia_idade                     0    Faixa 4 (47-57 anos)        0

     perfil_consumidor consumidor_natural_class
0                Misto                        1
1      Industrializado                        0
2              Saudável                       1
3                 Misto                       1
4                 Misto                       1
..                  ...                     ...
185               Misto                       1
186    Industrializado                        0
187            Saudável                       1
188               Misto                       1
189    Industrializado                        0

[190 rows x 25 columns]


# Salvar em CSV
df.to_csv("dataset_limpo.csv", index=False, sep=";")
print("Dados salvos em dataset_limpo.csv")
```

```
Dados salvos em dataset_limpo.csv
```

```
# Verificar valores ausentes
print(df.isnull().sum())

# Corrigir os NaNs em 'grupo_caminhada' para 'não'
df['grupo_caminhada'] = df['grupo_caminhada'].fillna('não')

# Corrigir os NaNs em 'profissao' para 'desconhecida'
df['profissao'] = df['profissao'].fillna('desconhecida')

# Corrigir os NaNs em 'autoavaliacao_saude' com o valor mais frequente (moda)
df['autoavaliacao_saude'] = df['autoavaliacao_saude'].fillna(df['autoavaliacao_saude'].mode()[0])

# Remover duplicatas
df.drop_duplicates(inplace=True)
```

```
id_cliente                        0
idade                             0
sexo                              0
escolaridade                      0
profissao                         0
autoavaliacao_saude               0
grupo_caminhada                   0
gasto_produtos_naturais           0
gasto_produtos_ultraprocessados   0
frequencia_compra                 0
participa_programa_saude          0
regiao                            0
participa_caminhada               0
profissao_encoded                 0
faixa_etaria_adulto               0
faixa_etaria_meia_idade           0
faixa_etaria_idoso                0
idade_zscore                      0
perfil_saudavel                   0
faixa_etaria                      0
alto_consumo_natural              0
faixa_etaria_otimizada            0
cluster                           0
perfil_consumidor                 0
consumidor_natural_class          0
dtype: int64
```

```
# Faixas etárias
df['faixa_etaria'] = pd.cut(df['idade'], bins=[0, 18, 35, 60, 100],
                            labels=['jovem', 'adulto', 'meia_idade', 'idoso'])

# Indicador binário de participação
df['participa_caminhada'] = df['grupo_caminhada'].apply(lambda x: 1 if x != 'não_participa' else 0)

print(df[['idade', 'faixa_etaria',  'participa_caminhada']])

df.head()
df['faixa_etaria'].value_counts()

print(df[['idade', 'faixa_etaria']].head())

df.to_csv("dados_atualizados.csv", index=False, encoding="utf-8")
```

```
     idade faixa_etaria  participa_caminhada
0       34       adulto                    1
1       28       adulto                    1
2       45   meia_idade                    1
3       52   meia_idade                    1
4       29       adulto                    1
..     ...          ...                  ...
185     35       adulto                    1
186     43   meia_idade                    1
187     34       adulto                    1
188     28       adulto                    1
189     53   meia_idade                    1

[190 rows x 3 columns]
   idade faixa_etaria
0     34       adulto
1     28       adulto
2     45   meia_idade
3     52   meia_idade
4     29       adulto
```

```
# Usando LabelEncoder
le = LabelEncoder()
df['profissao_encoded'] = le.fit_transform(df['profissao'])

# Recreate 'faixa_etaria' column
df['faixa_etaria'] = pd.cut(df['idade'], bins=[0, 18, 35, 60, 100],
                            labels=['jovem', 'adulto', 'meia_idade', 'idoso'])


# Ou usando One-Hot Encoding
df = pd.get_dummies(df, columns=['faixa_etaria'], drop_first=True)
print(df[['profissao', 'profissao_encoded', 'faixa_etaria_adulto', 'faixa_etaria_meia_idade', 'faixa_etaria_idoso']].head())
```

```
     profissao  profissao_encoded  faixa_etaria_adulto  faixa_etaria_adulto  \
0   professor                138                 True                 True
1    vendedor                180                 True                 True
2      médico                115                False                False
3  engenheiro                 61                False                False
4  secretária                153                 True                 True

   faixa_etaria_meia_idade  faixa_etaria_meia_idade  faixa_etaria_idoso  \
0                    False                    False               False
1                    False                    False               False
2                     True                     True               False
3                     True                     True               False
4                    False                    False               False

   faixa_etaria_idoso
0               False
1               False
2               False
3               False
4               False
```

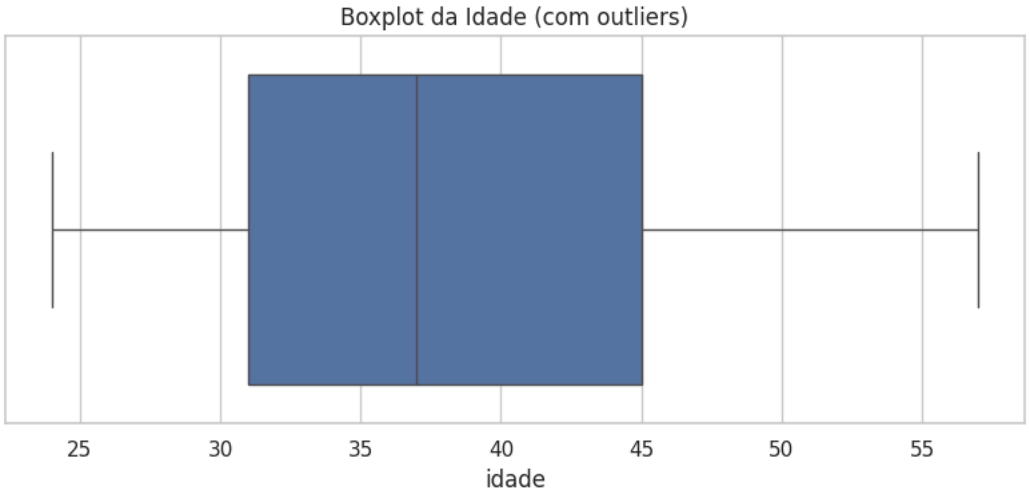
```
scaler = StandardScaler()
df['idade_zscore'] = scaler.fit_transform(df[['idade']])
print(df[['idade', 'idade_zscore']].head())
```

```
   idade  idade_zscore
0     34     -0.504597
1     28     -1.206109
2     45      0.781510
3     52      1.599941
4     29     -1.089191
```

```
# Detectar outliers sem alterar os dados originais
scaler = StandardScaler()
df['idade_zscore'] = scaler.fit_transform(df[['idade']])
outliers = df[np.abs(df['idade_zscore']) > 3]
print("Outliers identificados:")
print(outliers[['id_cliente', 'idade', 'idade_zscore']])
```

```
Outliers identificados:
Empty DataFrame
Columns: [id_cliente, idade, idade_zscore]
Index: []
```

```
plt.figure(figsize=(8, 4))
sns.boxplot(x=df['idade'])
plt.title('Boxplot da Idade (com outliers)', loc='center')
plt.tight_layout()  # Ajusta o layout para melhor centralização
plt.show()

# Salva o DataFrame atualizado
df.to_csv("dados_atualizados.csv", index=False, encoding="utf-8")

# index=False evita salvar a coluna de índice
# encoding="utf-8" evita problemas com acentos


#Fim da 2. Pré-processamento de Dados.
```

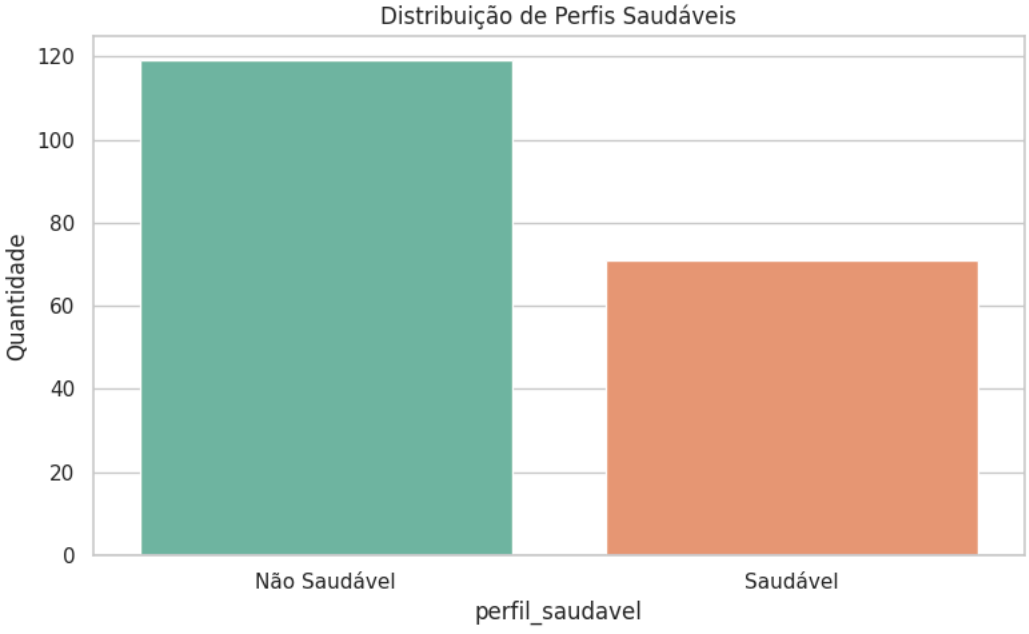



```
# 3. Análise Descritiva (EDA)


# Critérios para perfil saudável
limite_natural = df['gasto_produtos_naturais'].median()
limite_ultra = df['gasto_produtos_ultraprocessados'].median()

df['perfil_saudavel'] = (
    (df['gasto_produtos_naturais'] > limite_natural) &
    (df['gasto_produtos_ultraprocessados'] < limite_ultra) &
    (df['grupo_caminhada'] == 'Sim') &
    (df['autoavaliacao_saude'] >= 4)
).astype(int)

# Contar perfis saudáveis
total_saudaveis = df['perfil_saudavel'].sum()
print(f"Total de perfis com tendência saudável: {total_saudaveis}")
```

```
Total de perfis com tendência saudável: 71
```

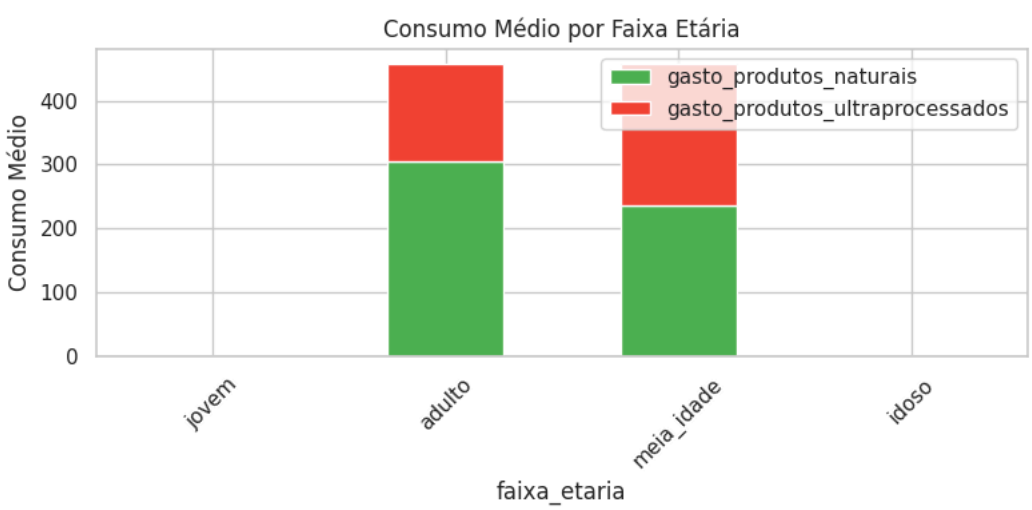
```
porcentagem = (total_saudaveis / len(df)) * 100
print(f"{porcentagem:.2f}% dos perfis têm tendência saudável")

# Visualização
sns.countplot(x='perfil_saudavel', data=df, palette='Set2')
plt.title("Distribuição de Perfis Saudáveis")
plt.xticks([0, 1], ['Não Saudável', 'Saudável'])
plt.ylabel("Quantidade")
plt.tight_layout()
plt.show()
```

```
37.37% dos perfis têm tendência saudável
```



Clique duas vezes (ou pressione "Enter") para editar

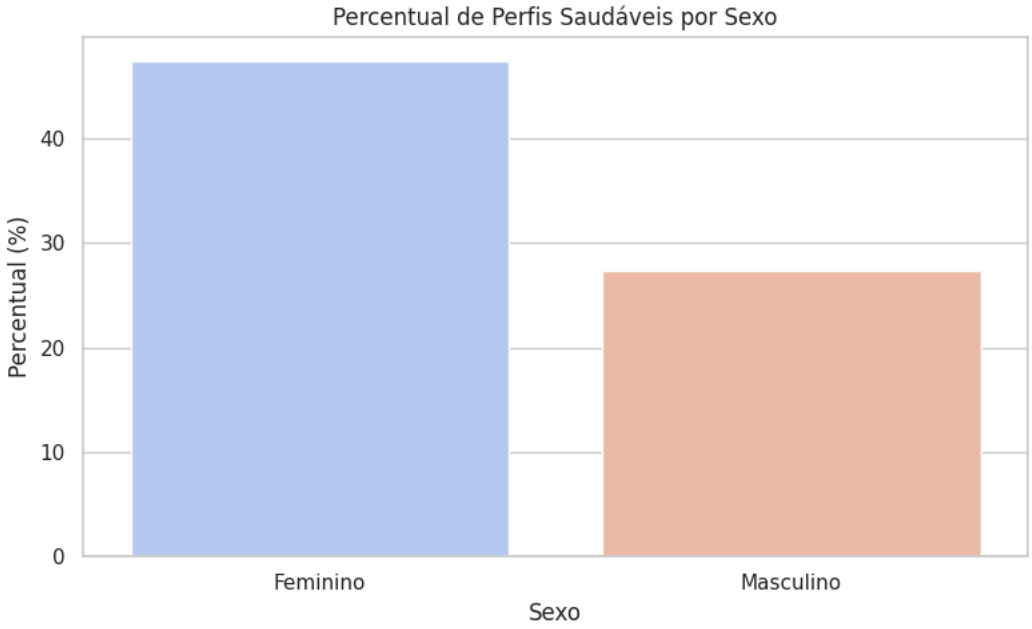
```
# Recreate the 'faixa_etaria' column to ensure it exists
df['faixa_etaria'] = pd.cut(df['idade'], bins=[0, 18, 35, 60, 100],
                            labels=['jovem', 'adulto', 'meia_idade', 'idoso'])

# Assuming 'gasto_produtos_naturais' is 'consumo_natural' and 'gasto_produtos_ultraprocessados' is 'consumo_ultraprocessado'
df_grouped = df.groupby('faixa_etaria')[['gasto_produtos_naturais', 'gasto_produtos_ultraprocessados']].mean().reset_index()

df_grouped.plot(x='faixa_etaria', kind='bar', stacked=True, figsize=(8,4), color=['#4CAF50', '#F44336'])
plt.title('Consumo Médio por Faixa Etária')
plt.ylabel('Consumo Médio')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



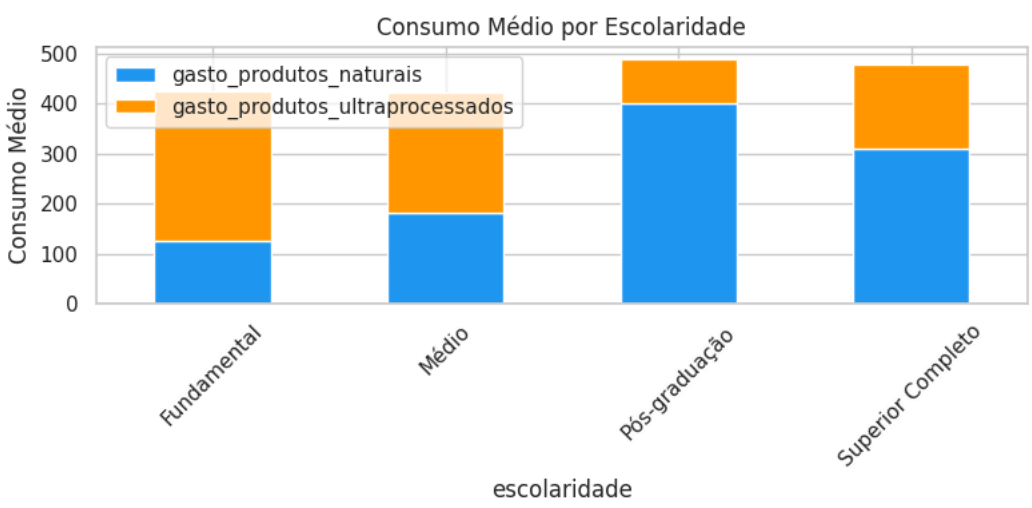

```
# Contagem de perfis saudáveis por sexo
saudaveis_por_sexo = df.groupby('sexo')['perfil_saudavel'].sum()
print(saudaveis_por_sexo)
```

```
sexo
Feminina      1
Feminino     44
Masculino    26
Name: perfil_saudavel, dtype: int64
```

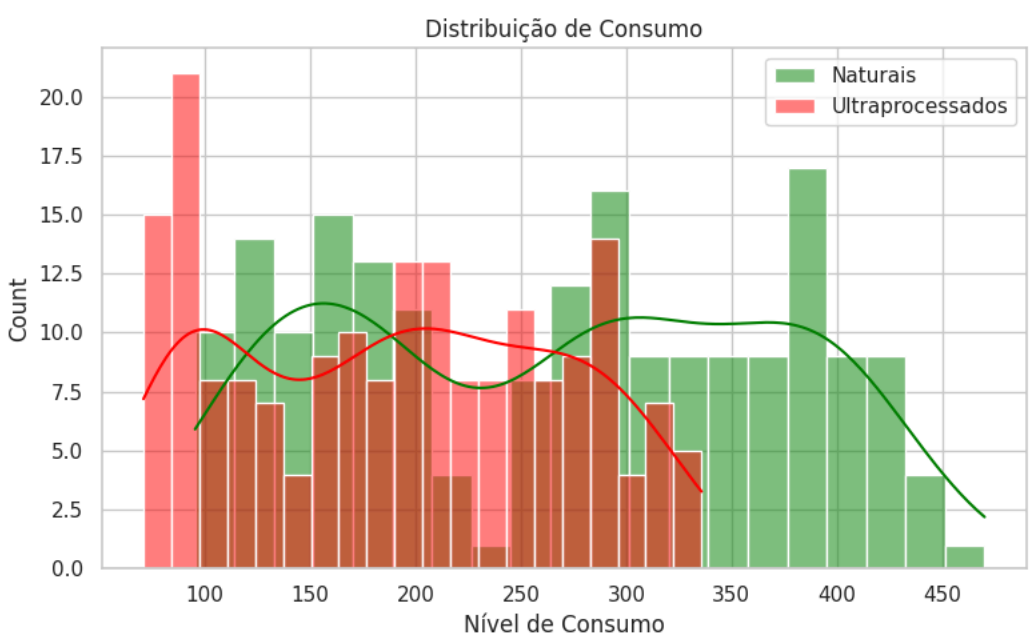
```
# Padronizar os valores da coluna 'sexo'
df['sexo'] = df['sexo'].replace({
    'Feminina': 'Feminino',   # corrige Feminina para Feminino
    'Masculino': 'Masculino',
    'Feminino': 'Feminino'
})

# Agora refaça os cálculos
saudaveis_por_sexo = df.groupby('sexo')['perfil_saudavel'].sum()
total_por_sexo = df['sexo'].value_counts()
percentual_saudaveis = (saudaveis_por_sexo / total_por_sexo) * 100

# Gráfico
sns.barplot(x=percentual_saudaveis.index, y=percentual_saudaveis.values, hue=percentual_saudaveis.index, palette='coolwarm', legend=Fals
plt.title("Percentual de Perfis Saudáveis por Sexo")
plt.ylabel("Percentual (%)")
plt.xlabel("Sexo")
plt.tight_layout()
plt.show()
```




```
df_edu = df.groupby('escolaridade')[['gasto_produtos_naturais', 'gasto_produtos_ultraprocessados']].mean().reset_index()

df_edu.plot(x='escolaridade', kind='bar', stacked=True, figsize=(8,4), color=['#2196F3', '#FF9800'])
plt.title('Consumo Médio por Escolaridade')
plt.ylabel('Consumo Médio')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(8,5))
sns.histplot(df['gasto_produtos_naturais'], kde=True, color='green', label='Naturais', bins=20)
sns.histplot(df['gasto_produtos_ultraprocessados'], kde=True, color='red', label='Ultraprocessados', bins=20)
plt.title('Distribuição de Consumo')
plt.xlabel('Nível de Consumo')
plt.legend()
plt.tight_layout()
plt.show()
```

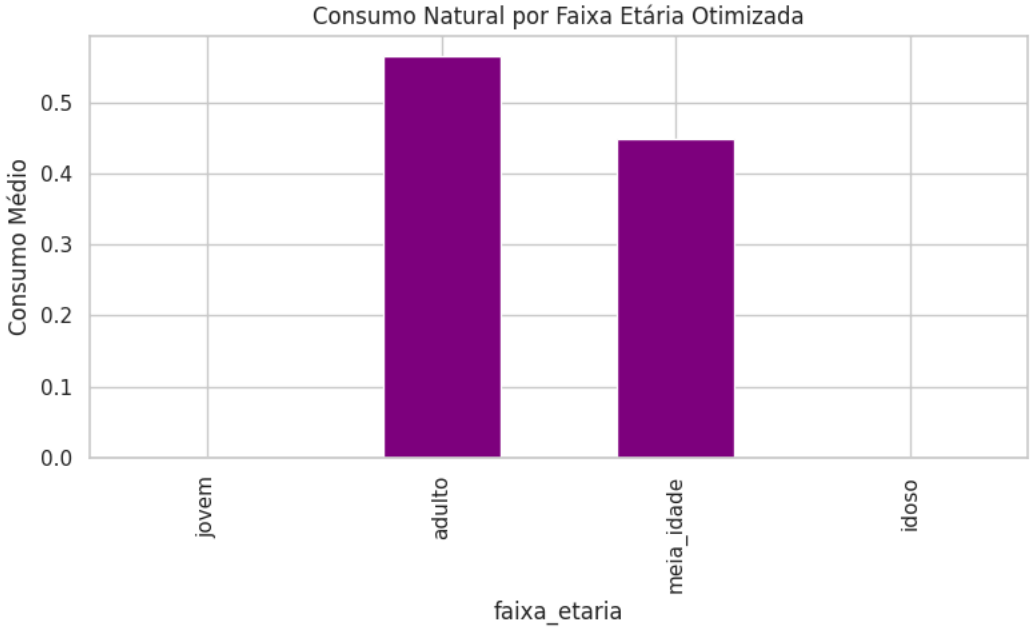



```
# Criar variável binária: alto consumo natural
limite = df['gasto_produtos_naturais'].median()
df['gasto_produtos_naturais'] = (df['gasto_produtos_naturais'] > limite).astype(int)

# Tabela de contingência
contingencia = pd.crosstab(df['profissao'], df['gasto_produtos_naturais'])
print(contingencia)

# Salvar CSV
contingencia.to_csv('contingency_table.csv')
```

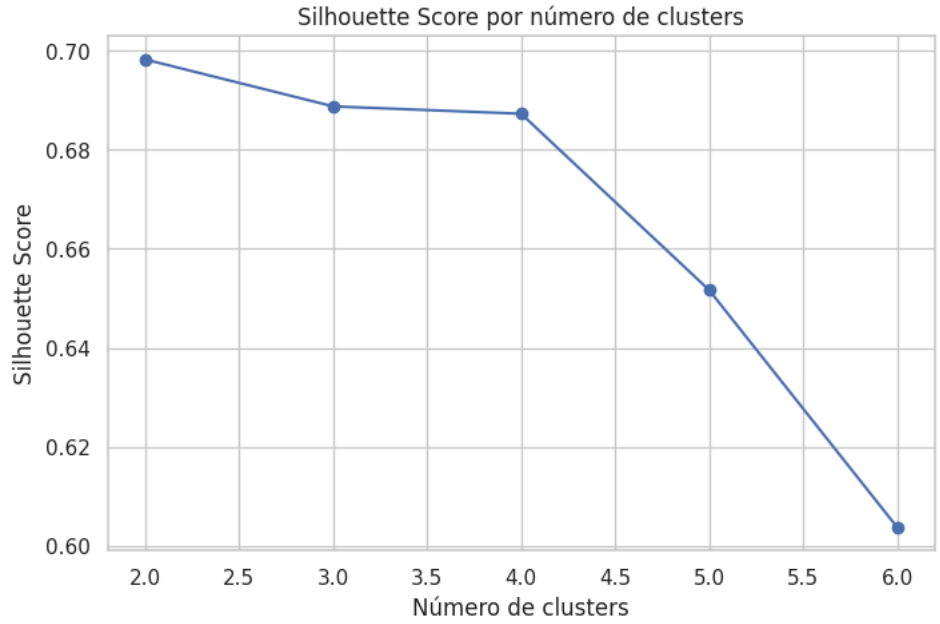
```
gasto_produtos_naturais  0  1
profissao
acupunturista            0  1
administradora           0  1
ajudante                 1  0
ajustador                1  0
analista                 1  0
...                     .. ..
veterinária              0  1
vigilante                1  0
virologista              0  1
web designer             1  0
zeladora                 1  0

[187 rows x 2 columns]
```

```
# Criar variável binária como target
target = (df['gasto_produtos_naturais'] > df['gasto_produtos_naturais'].median()).astype(int)

# Treinar árvore para encontrar cortes de idade
tree = DecisionTreeClassifier(max_leaf_nodes=4)
tree.fit(df[['idade']], target)

# Obter cortes
thresholds = np.sort(tree.tree_.threshold[tree.tree_.threshold > 0])
print("Cortes sugeridos:", thresholds)

# Criar faixas etárias guiadas
bins = [0] + list(thresholds) + [df['idade'].max()]
labels = [f'Faixa {i+1}' for i in range(len(bins)-1)]
df['faixa_etaria_otimizada'] = pd.cut(df['idade'], bins=bins, labels=labels)

# Visualizar consumo por faixa otimizada
df.groupby('faixa_etaria')['gasto_produtos_naturais'].mean().plot(kind='bar', color='purple', figsize=(8,5))
plt.title('Consumo Natural por Faixa Etária Otimizada')
plt.ylabel('Consumo Médio')
plt.tight_layout()
plt.show()
```

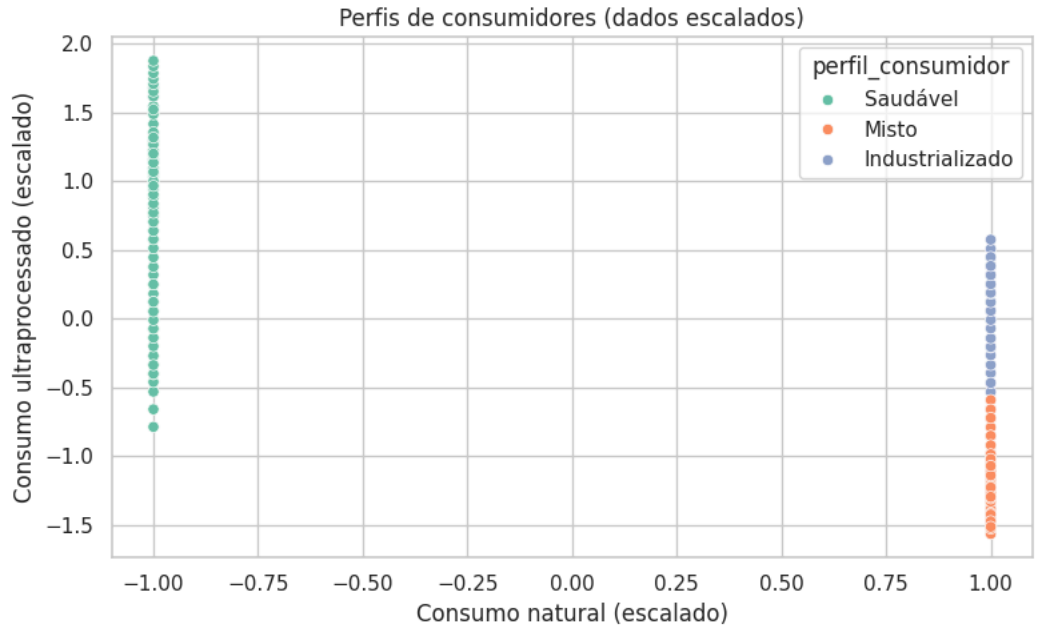
```
Cortes sugeridos: [30.5 39.5 47.5]
```




```
# 4. Agrupamento (Clustering) e Classificação


# Selecionar as variáveis para clustering
X = df[['gasto_produtos_naturais', 'gasto_produtos_ultraprocessados']]

# Padronizar os dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Testar diferentes números de clusters com Silhouette Score
silhouettes = []
for k in range(2, 7):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_scaled)
    silhouettes.append(silhouette_score(X_scaled, labels))

# Plotar Silhouette Score
plt.plot(range(2, 7), silhouettes, marker='o')
plt.title("Silhouette Score por número de clusters")
plt.xlabel("Número de clusters")
plt.ylabel("Silhouette Score")
plt.show()

# Escolher o número ideal de clusters (exemplo: 3)
k_opt = 3
kmeans = KMeans(n_clusters=k_opt, random_state=42, n_init=10)
df['cluster'] = kmeans.fit_predict(X_scaled)

# Ver médias por cluster para entender os perfis
print("\nMédias por cluster:")
print(df.groupby('cluster')[['gasto_produtos_naturais', 'gasto_produtos_ultraprocessados']].mean())

# Nomear os clusters com base nas médias observadas
cluster_names = {
    0: 'Industrializado',
    1: 'Saudável',
    2: 'Misto'
}
df['perfil_consumidor'] = df['cluster'].map(cluster_names)

# Visualização dos clusters com nomes
sns.scatterplot(x=X_scaled[:, 0], y=X_scaled[:, 1], hue=df['perfil_consumidor'], palette='Set2')
plt.title("Perfis de consumidores (dados escalados)")
plt.xlabel("Consumo natural (escalado)")
plt.ylabel("Consumo ultraprocessado (escalado)")
plt.tight_layout()
plt.show()

# Contagem de consumidores por perfil
print("\nContagem por perfil:")
print(df['perfil_consumidor'].value_counts())
```




```
Médias por cluster:
         gasto_produtos_naturais  gasto_produtos_ultraprocessados
cluster
0                            1.0                       196.694118
1                            0.0                       248.601579
2                            1.0                        99.167213
```

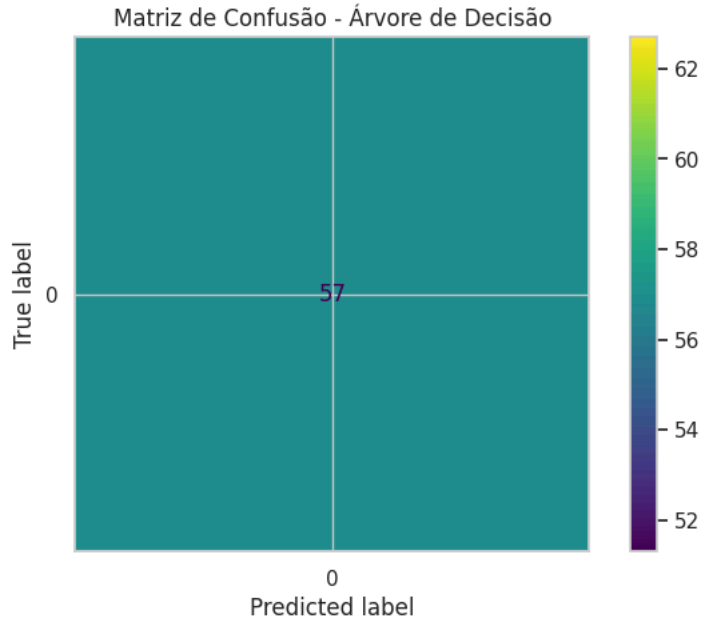


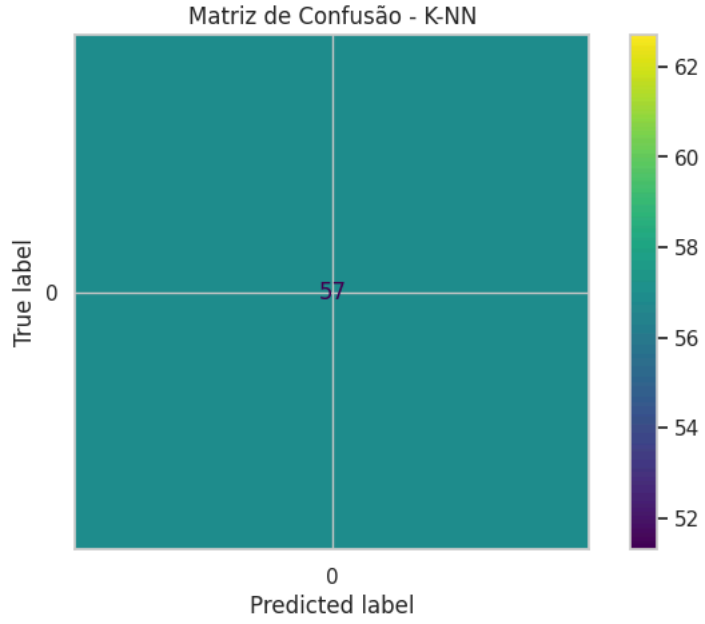

```
Contagem por perfil:
perfil_consumidor
Saudável          95
Misto             61
Industrializado   34
Name: count, dtype: int64
```

```
# Criar variável alvo: consumidor natural (gasto natural > ultraprocessado)
df['consumidor_natural'] = (df['gasto_produtos_naturais'] > df['gasto_produtos_ultraprocessados']).astype(int)

# Selecionar variáveis preditoras
features = ['idade', 'sexo', 'escolaridade', 'autoavaliacao_saude', 'grupo_caminhada', 'frequencia_compra']
X = df[features]
y = df['consumidor_natural']

# Convert categorical features to numerical
X = pd.get_dummies(X, columns=['sexo', 'escolaridade', 'grupo_caminhada'], drop_first=True)



# Dividir em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Árvore de Decisão
```

```
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X_train, y_train)
y_pred_tree = tree.predict(X_test)

# K-NN
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

# Avaliação
acc_tree = accuracy_score(y_test, y_pred_tree)
acc_knn = accuracy_score(y_test, y_pred_knn)

print(f"Acurácia Árvore de Decisão: {acc_tree:.2f}")
print(f"Acurácia K-NN: {acc_knn:.2f}")

# Matrizes de confusão
ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred_tree)).plot()
plt.title("Matriz de Confusão - Árvore de Decisão")
plt.show()

ConfusionMatrixDisplay(confusion_matrix(y_test, y_pred_knn)).plot()
plt.title("Matriz de Confusão - K-NN")
plt.show()
```

```
Acurácia Árvore de Decisão: 1.00
Acurácia K-NN: 1.00
```






```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

```
# Carregar os dados
df = pd.read_csv("/content/drive/MyDrive/dataset_dados.csv", sep=';')
df = df.drop_duplicates()

# Criar colunas binárias para categorias de gasto
df['compra_naturais'] = (df['gasto_produtos_naturais'] > 250).astype(int)
df['compra_ultraprocessados'] = (df['gasto_produtos_ultraprocessados'] > 200).astype(int)

# Simular outras categorias de produtos (exemplo)
df['compra_snacks_saudaveis'] = ((df['gasto_produtos_naturais'] > 300) & (df['autoavaliacao_saude'] >= 4)).astype(int)
df['compra_orgânicos'] = ((df['gasto_produtos_naturais'] > 350) & (df['grupo_caminhada'] == 'Sim')).astype(int)

# Selecionar apenas colunas binárias
basket = df[['compra_naturais', 'compra_ultraprocessados', 'compra_snacks_saudaveis', 'compra_orgânicos']]

print(basket.head())
```

```
   compra_naturais  compra_ultraprocessados  compra_snacks_saudaveis  \
0                1                        0                        0
1                0                        1                        0
2                1                        0                        1
3                1                        0                        0
4                0                        0                        0

   compra_orgânicos
0                 0
1                 0
2                 0
3                 0
4                 0
```

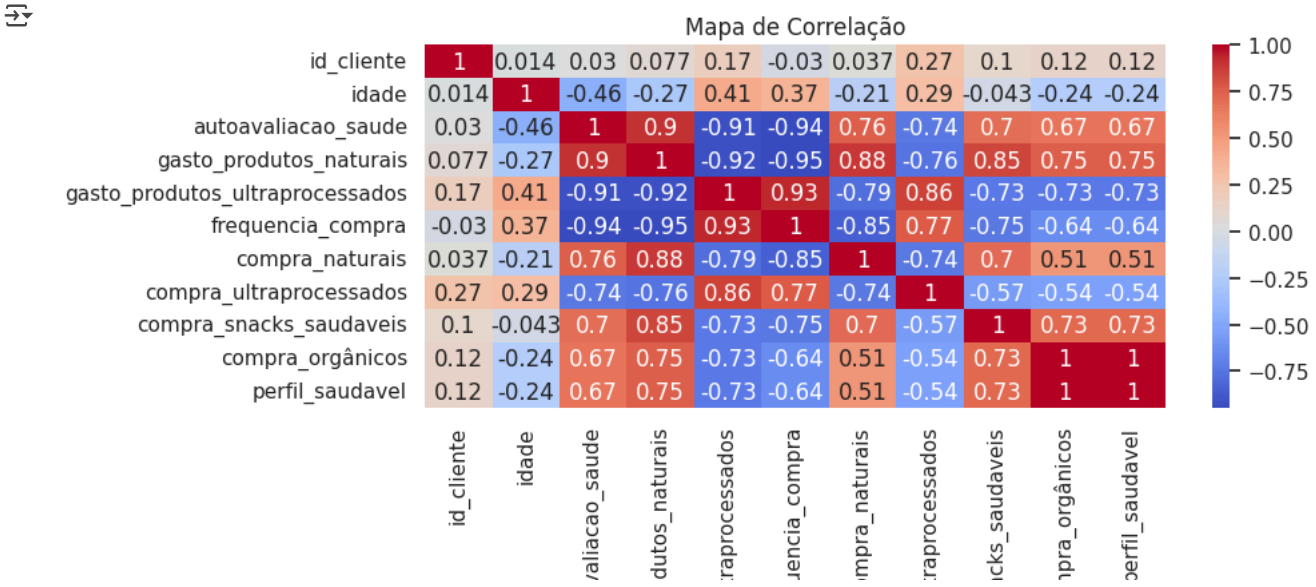
```
# Gerar itemsets frequentes
frequent_itemsets = apriori(basket, min_support=0.1, use_colnames=True)

# Gerar regras de associação
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0)

# Filtrar regras fortes
strong_rules = rules[(rules['confidence'] >= 0.6) & (rules['lift'] >= 1.2)]

# Exibir regras relevantes
print(strong_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
                                 antecedents  \
0                    (compra_snacks_saudaveis)
1                            (compra_naturais)
2                           (compra_orgânicos)
4                    (compra_snacks_saudaveis)
5                           (compra_orgânicos)
6   (compra_snacks_saudaveis, compra_orgânicos)
7    (compra_snacks_saudaveis, compra_naturais)
8           (compra_orgânicos, compra_naturais)
9                    (compra_snacks_saudaveis)
10                          (compra_orgânicos)

                                consequents   support  confidence      lift
0                           (compra_naturais)  0.410526    1.000000  1.711712
1                    (compra_snacks_saudaveis)  0.410526    0.702703  1.711712
2                           (compra_naturais)  0.268421    1.000000  1.711712
4                           (compra_orgânicos)  0.268421    0.653846  2.435897
5                    (compra_snacks_saudaveis)  0.268421    1.000000  2.435897
6                           (compra_naturais)  0.268421    1.000000  1.711712
7                           (compra_orgânicos)  0.268421    0.653846  2.435897
8                    (compra_snacks_saudaveis)  0.268421    1.000000  2.435897
9           (compra_orgânicos, compra_naturais)  0.268421    0.653846  2.435897
10  (compra_snacks_saudaveis, compra_naturais)  0.268421    1.000000  2.435897
```

```
# Verificar clientes que aparecem em múltiplas regras
df['perfil_saudavel'] = (df['compra_orgânicos'] & df['compra_snacks_saudaveis']).astype(int)
perfil_count = df['perfil_saudavel'].sum()
print(f"Número de clientes com perfil saudável recorrente: {perfil_count}")
```

```
Número de clientes com perfil saudável recorrente: 51
```

```
# Mapa de calor das correlações
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Mapa de Correlação")
plt.tight_layout()
plt.show()
```


Mapa de Correlação
id_cliente 1 0.014 0.03 0.077 0.17 -0.03 0.037 0.27 0.1 0.12 0.12
idade 0.014 1 -0.46 -0.27 0.41 0.37 -0.21 0.29 -0.043 -0.24 -0.24
autoavaliacao_saude 0.03 -0.46 1 0.9 -0.91 -0.94 0.76 -0.74 0.7 0.67 0.67
gasto_produtos_naturais 0.077 -0.27 0.9 1 -0.92 -0.95 0.88 -0.76 0.85 0.75 0.75
gasto_produtos_ultraprocessados 0.17 0.41 -0.91 -0.92 1 0.93 -0.79 0.86 -0.73 -0.73 -0.73
frequencia_compra -0.03 0.37 -0.94 -0.95 0.93 1 -0.85 0.77 -0.75 -0.64 -0.64
compra_naturais 0.037 -0.21 0.76 0.88 -0.79 -0.85 1 -0.74 0.7 0.51 0.51
compra_ultraprocessados 0.27 0.29 -0.74 -0.76 0.86 0.77 -0.74 1 -0.57 -0.54 -0.54
compra_snacks_saudaveis 0.1 -0.043 0.7 0.85 -0.73 -0.75 0.7 -0.57 1 0.73 0.73
compra_orgânicos 0.12 -0.24 0.67 0.75 -0.73 -0.64 0.51 -0.54 0.73 1 1
perfil_saudavel 0.12 -0.24 0.67 0.75 -0.73 -0.64 0.51 -0.54 0.73 1 1
id_cliente
idade
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75