jupyter SegmentacaoClientes (autosaved) (conclusão(s) ausente(s)) ainda - dockerized

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Segmentação de Clientes

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
```

In [2]:
```
df_ecom_full = pd.read_csv('./BaseOlistEcom.zip')
```

In [3]:
```
df_ecom.info()
```

```
<class> pandas.core.frame.DataFrame>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 19):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   seq_id                 112650 non-null  int64
 1   order_id               112650 non-null  object
 2   id_order_item          112650 non-null  int64
 3   id_product             112650 non-null  object
 4   id_seller              112650 non-null  object
 5   shipping_limit         112650 non-null  datetime64[ns]
 6   estado_cliente         112650 non-null  object
 7   id_cliente             112650 non-null  object
 8   status_pedido          112650 non-null  object
 9   data_pedido            112650 non-null  datetime64[ns]
 10  pedido_aprovado        112557 non-null  datetime64[ns]
 11  data_pedido_enviado    110197 non-null  datetime64[ns]
 12  pedido_entregue        108845 non-null  datetime64[ns]
 13  estimativa_entrega     112650 non-null  datetime64[ns]
 14  id_cliente             112650 non-null  object
 15  preco                  112650 non-null  float64
 16  frete                  112650 non-null  float64
 17  nome_produto           111047 non-null  object
 18  segmento_loja          112650 non-null  object
 19  categoria_produto      111047 non-null  object
 20  segmento_vitrine       112650 non-null  object
 21  review                 112650 non-null  int64
 22  id_review              112650 non-null  object
 23  categoria_review       112650 non-null  object
 24  id_cidade              112650 non-null  int64
 25  cidade_cendec          112650 non-null  object
 26  estado_cendec          112650 non-null  object
dtypes: float64(2), int64(6), object(17)
memory usage: 23.5 MB
```

In [4]:
```
df_ecom[0]
```

	pesquisa	id_cliente	id_order	id_order_item	id_product	id_seller	estado_cliente
0	0	3504f0f0d4ac3e7e7e2d0d0d0d0d0d	00246f17a0a0a0a0a0a0a0a0a0a0a0	1	1e4e8f	4f4	SAO PAULO/SP/BRASIL/BRASIL/SP
1	1	3010f2c3d4e7e8f0d0d1e1f2e1f2e1	3ee4e3d0d0a0a0a0a0a0a0a0a0a0a0	1	000f00	aaa1aaa1	PA, 2.1E2E2E4E4E2E2E2E2E2E2E
2	1	3010f2c3d4e7e8f0d0d1e1f2e1f2e1	3ee4e3d0d0a0a0a0a0a0a0a0a0a0a0	2	000f00	aaa1aaa1	PA, 1.1E2E2E4E4E2E2E2E2E2E2E
3	2	7b4c9e2a1d7f8b0c7e4d1f3a9b2c6e	1a7f9c0e1b2d3a4f5b6c7d8e9f0a1b	1	5b4d00	4ccc4ccc	PE, 1.1E2E2E4E4E2E2E2E2E2E2E
4	3	9c2d4f6a8b1e3c5d7f9a0b2c4e6f8a	2c5d7f9a0b1e3c5d7f9a0b2c4e6f8a	1	6a0000	4ccc4ccc	SP, 1.1E2E2E4E4E2E2E2E2E2E2E

5 rows × 27 colunas

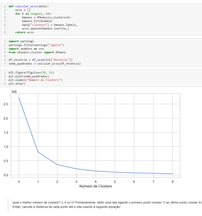
Análise Exploratória / Levantar hipóteses

In [5]:
```
%matplotlib inline
plt.figure(figsize=(8,6))

df = df_ecom.groupby('estado_cliente', sort=False)
df['id_cliente'].count().plot(kind='bar')
plt.title('Distribuição de estado dos clientes', fontsize=18)
plt.xlabel('Estado', fontsize=12)
plt.ylabel('Clientes', fontsize=12)
plt.ticks(rotation=0, ha='center')
plt.show()
for i in p.patches:
    p.annotate(str(i.get_height()),
               (i.get_x(), i.get_height()+
                int(i.get_height()/2.5),
                fontsize=10,
                rotation=90,
                color='white')
plt.savefig('estado_cliente.png')
```

In [6]:
```
plt.savefig('estado_cliente.png')
```

In [7]:
```
%matplotlib inline
plt.figure(figsize=(8,6))

df = df_ecom.groupby('estado_cliente', sort=False)
df['id_cliente'].count().plot(kind='bar')
plt.title('Distribuição de estado dos clientes', fontsize=18)
plt.xlabel('Estado', fontsize=12)
plt.ylabel('Clientes', fontsize=12)
plt.ticks(rotation=0, ha='center')
plt.show()
```

In [8]:
```
plt.savefig('segmento_loja.png')
```



In [9]:
```
plt.savefig('segmento_loja.png')
```

- A maioria dos clientes tendia comprar por cartão de crédito, posteriormente por boleto, voucher e cartão de débito. Seria interessante averiguar as promoções para pagamento a vista (para aumento dos pagamentos com boleto ou cartão de débito).
- A maioria dos vendedores e compradores estão em SP (não estava acontecendo nem nosso último trimestre quando a atrasgdência do seu produto).

Concluída data wrangling, vague conceitualização de um novo problema, vamos ganhar um conhecimento que vou precisar para fazer a análise de textbook, frequência e recência.

In [10]:
```
df_clean = df[['id_order','id_cliente','data_pedido', 'data_pedido', 'preco']]
df_clean.sample(5).sort_values('id_cliente')
```

In [11]:
```
df_clean['DaysSinceLast'] = pd.to_datetime(df_clean['DaysSinceLast'])
```

Precisamos definir agora as três variáveis que utilizaremos para segmentação: **recência, frequência e receita**. Método utilizado para embarcar o comportamento de compra de cada cliente e também segmentar os clientes, também pode ajudar a entender melhor os grupos com suas chas de compra.

- Recência: ação recentemente um cliente fez uma compra. É um importante predito para um cliente que comprou há pouco tempo tem mais chance de voltar a comprar do que aquele que comprou há muito tempo. Uma ação importante seria promoções específicas para responder a promoções.
- Frequência: com que frequência um cliente faz uma compra. Se um cliente compra frequentemente, espera-se que ele volte a comprar.
- Receita: quanto um cliente gastou em uma ocasião. Aquelles clientes que gastaram muito tem mais valor para a organização em comparação com aqueles que gastaram pouco.

In [12]:
```
# Monta df observação (uma p ada matriz de compra no ano)
df_clean['DaysSinceLast'] = (df_clean['DaysSinceLast'] - df_clean['DaysSinceLast'])/np.timedelta64(1,'D')

# Ordena essa df com a nossa nova chamada df_user
df_user = df_clean[df_clean.id_cliente].sort_values(['id_cliente_cliente', 'Recencia'])[['id_cliente_cliente']]

df_user[0:5].head()
```

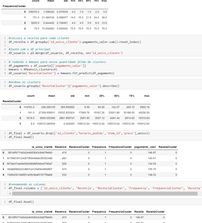
In [13]:

	id_cliente_cliente	id_cliente	recencia_pedido	data_ok	preco	Recencia
0	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	2017-05-30 00:00:00	1.0	150.00	273
1	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	0a1b1c2d3e4f5a6b7c8d9e0f1a2b3c	2017-05-30 00:00:00	1.0	110.00	273
2	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	0b1b2c3d4e5f6a7b8c9d0e1f2a3b4c	2017-05-30 00:00:00	2.0	132.90	256
3	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c	2017-05-30 00:00:00	1.0	110.00	256
4	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d	2017-05-30 00:00:00	1.0	150.00	256

In [14]:
```
# calcular pontuações
df = df_user
for i in range(1, 4):
    print('Processando', i)
    qseq = df[['id_cliente', i]].sort_values(i)
    qcut = pd.qcut(qseq[i], 5, labels=False)
    df = df.sort_values('id_cliente')
    df[str(i)+'_score'] = qcut.fillna(0).astype(int).values
    df.head(3)
```

In [15]:
```
import sqlalchemy
import sqlalchemy.ext.sqlite
import numpy as np
%matplotlib inline
df_recencia = df_user[['Recencia']]
sns_pairplot = sns.pairplot(df_recencia)
```

In [16]:
```
plt.figure(figsize=(8, 5))
plt.plot(sns_pairplot)
plt.xlabel('Número de Clusters')
plt.show()
```



Qual o melhor número de clusters? 3 ou 10 Primeiramente, definir uma nova regra para o primeiro ponto (cluster 1) ao último ponto (cluster 5). Então, calcular a distância de cada ponto até a reta usando a seguinte equação:

$$\text{distance}(P_0, P_1, (x, y)) = \frac{|(y_1 - y_0)x - (x_1 - x_0)y + x_1 y_0 - y_1 x_0|}{\sqrt{(y_1 - y_0)^2 + (x_1 - x_0)^2}}$$

Nesta caso, P0 é o ponto do cluster 1 e P1 o ponto do cluster 5. O que se interpreta as coordenadas de qualquer ponto que se quiere calcular a distância. Calculando todas as distâncias, consegue determinar o ponto ideal

In [17]:
```
import math
df_sqrt_clusters_clusterclass=[]
SQ_df = 0.0 - sum(df)
for i in range(0, 5):
    print('.')
    distancias = []
    for k in range(clusters):
    x = df[i]
    y = df[i]
    x1 = df[i+1]
    y1 = df[i+1]
    denominador = math.sqrt((x1 - x)**2 + (y1 - y)**2)
    distancias.append(numerador/denominador)
    return distancias.index(max(distancias)) + 1

# Calculando a soma dos quadrados para as quantidades de clusters
df_sqrt_clusters = df[['id_cliente']].sample(5)
df_sqrt_clusters.head()

# Calculando a quantidade ótima de clusters
df_sqrt_clusters['cluster_ideal'] = df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()
df_sqrt_clusters[0:5].head()
```

In [18]:
```
# Agora o número ideal de clusters
df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
```

In [19]:
```
# Ordenando a nova df com a nova quantidade ótima de clusters
df = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
Recencia = df[['Recencia']]
Recencia.sample(5).sort_values('Recencia')
```

In [20]:
```
# Ordenando a nova df com a nova quantidade ótima de clusters
df = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
Frequencia = df[['Frequencia']]
Frequencia.sample(5).sort_values('Frequencia')
```

In [21]:
```
# Ordenando a nova df com a nova quantidade ótima de clusters
df = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
Receita = df[['Receita']]
Receita.sample(5).sort_values('Receita')
```

In [22]:
```
# Ordenando a nova df com a nova quantidade ótima de clusters
df = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_segmento = df_sqrt_clusters[['segmento_cliente']]
Mesmo = df[['id_cliente']]
df_final = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_final.head()
```

In [23]:
```
# Ordenando a nova df com a nova quantidade ótima de clusters
df = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_segmento = df_sqrt_clusters[['segmento_cliente']]
Mesmo = df[['id_cliente']]
df_final = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_final.head()
```

In [24]:
```
# Ordenando a nova df com a nova quantidade ótima de clusters
df = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_segmento = df_sqrt_clusters[['segmento_cliente']]
Mesmo = df[['id_cliente']]
df_final = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_final.head()
```

In [25]:
```
# Ordenando a nova df com a nova quantidade ótima de clusters
df = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_segmento = df_sqrt_clusters[['segmento_cliente']]
Mesmo = df[['id_cliente']]
df_final = df_sqrt_clusters[df_sqrt_clusters[lambda x: x.cluster_ideal].count_index()]
df_final.head()
```

In [26]:
```
df_final['Recencia'] = df_final['Recencia'].astype(int)
df_final['Frequencia'] = df_final['Frequencia'].astype(int)
df_final['Receita'] = df_final['Receita'].astype(int)
df_final.head()
```

	id_cliente_cliente	Recencia	Recencia1cluster	Frequencia	Frequencia1cluster	Receita	Receita1cluster
0	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	273	0	1	0	150.00	0
1	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	273	0	1	0	110.00	0
2	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	256	0	2	0	132.90	0
3	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	256	0	1	0	110.00	0
4	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	256	0	1	0	150.00	0

In [27]:
```
df_final['Recencia'] = df_final['Recencia1cluster']
df_final['Frequencia'] = df_final['Frequencia1cluster']
df_final['Receita'] = df_final['Receita1cluster']
df_final.head()
```

Calcularemos a pontuação geral do cliente baseado nesses clusters e também definiremos 4 opções: ouro, bronze, ouro e premium.

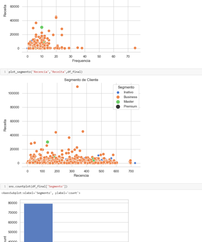
In [28]:
```
df_final['Pontuacao'] = df_final['Recencia1cluster'] + df_final['Frequencia1cluster'] + df_final['Receita1cluster']
df_final.groupby('Pontuacao').Recencia.'Frequencia','Receita','Pontuacao'].mean()
df_final.head()
```

	id_cliente_cliente	Recencia	Recencia1cluster	Frequencia	Frequencia1cluster	Receita	Receita1cluster	Pontuacao
0	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	273	0	1	0	150.00	0	0
1	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	273	0	1	0	110.00	0	0
2	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	256	0	2	0	132.90	0	0
3	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	256	0	1	0	110.00	0	0
4	0a4d4d4d4d4d4d4d4d4d4d4d4d4d4d	256	0	1	0	150.00	0	0

In [29]:
```
df_final['segmento'] = 'Ouro'

df_final.loc[df_final['Pontuacao'].isin, 'segmento'] = 'Bronze'
df_final.loc[df_final['Pontuacao'].isin, 'segmento'] = 'Ouro'
df_final.loc[df_final['Pontuacao'].isin, 'segmento'] = 'Premium'

df_final.sample(10).sort
```

In [30]:
```
df_final.loc[df_final['Pontuacao'].isin, 'segmento'] = 'Bronze'
df_final.loc[df_final['Pontuacao'].isin, 'segmento'] = 'Ouro'
df_final.loc[df_final['Pontuacao'].isin, 'segmento'] = 'Premium'
df_final.head()
```

In [31]:
```
df_segmento = df_final[['Recencia', 'Frequencia', 'segmento']]
sns_segmento = sns.pairplot(df_segmento, dataset=df_segmento, hue='segmento', size=2.5, size_prefer='Frequencia', 'Recencia', 'Frequencia', 'segmento')
sns.set_title('Segmento de Cliente')
plt.show()
```



In [32]:
```
plt.savefig('segmento_cliente.jpg')
```

In [33]:
```
plt.savefig('segmento_cliente.jpg')
```


In [34]:
```
sns.countplot(df_final['segmento'])
```

Out[34]:
```
<matplotlib.axes.subplots.AxesSubplot>
```


Método utilizado de clusterização. É menos, acima descrição detalhada de como determinar matematicamente o melhor número de clusters. Por fim, consegui dividir os clientes com base no comportamento. Em seguida criar outra figura de segmentação com base nas características de cada cluster. Por exemplo, clientes que compram muito e que compram há pouco tempo são clientes ouro. Clientes que compram muito, mas com pontuação ruim, são clientes que compram muito, mas com pouca frequência (também ardem bronze) e uma pontuação semelhante aos que gastam menos, mas com mais frequência. Para bater uma análise mais complexa, criei uma tabela com base em uma pontuação de 1 a 5 (sino o exemplo abaixo):

Segmentos	Descrição	Recencia	Frequencia	Receita
Campeões	Compraram recentemente, compram com frequência e gastam muito	4-5	4-5	4-5
Fieis	Compram com frequência. Responsáveis, compartilhamos valor	3-5	4-5	3-5
Potencialmente fieis	Compradores recentemente, mas ainda com pouca quantidade	4-5	2-3	3-5
Novos clientes	Compraram recentemente, mas não compram com frequência	4-5	0-1	0-1
Promissores	Compraram recentemente, mas não gastaram muito	3-4	0-1	0-1
Precisam de atenção	Recência, frequência e valor monetário abaixo da média	2-3	2-3	2-3
Em risco	Gastavam muito, compravam muitas vezes, mas há muito tempo	1-2	4-5	4-5
Não pode perder ao	Compravam muito e por muito tempo, mas há muito tempo	1-2	4-5	4-5
Hibernando	Compraram pouco, com baixa frequência e há muito tempo	1-2	2-3	2-3
Perdidos	Recência, frequência e valor monetário baixos	1-2	1-2	1-2

In [35]:

In [36]:

In [37]:

In [38]:

In [39]:

In [40]: