Centro Universitário Leonardo da Vinci – UNIASSELVI
Curso Big Data e Ciência Analítica (FLD209127CET) – Imersão Profissional: Futuro dos Dados e Aplicação de Big Data

CLUSTERING EM E-COMMERCE: SEGMENTAÇÃO DE CLIENTES PARA ORIENTAÇÃO DE MARKETING DIRECIONADO

Autor: Guilherme Bibiano Santos[1]

Tutor externo: Guilherme Augusto Reis dos Santos[2]

MOTIVO DA ESCOLHA DO OBJETO DE ESTUDO

E-commerce é um ramo muito comum para pequenos e grandes empresários e o Big data abrange ambos. Aprender a manusear os respectivos tipos de informações e entender o contexto do negócio para resolução dos problemas se torna uma habilidade essencial para um cientista de dados.

É um mercado de proporções bilionárias, sejam estes promovidos por multinacionais ou pequenos e médios empreendedores nacionais, a demanda por estratégias baseadas em dados se torna uma cultura necessária para reter os ganhos e minimizar os custos operacionais.

Este objeto de estudo abrange a segmentação de clientes para direcionar campanhas personalizadas (ofertas, assinaturas, produtos, exclusividades) para o cliente correto em tempo hábil, assim, evitando a perda da fidelização deles e maximizando a monetização com os melhores clientes.

ESTRATÉGIAS DE ANÁLISE DO OBJETO

A estratégia adotada neste projeto fundamenta-se na aplicação de algoritmos de clustering (de aprendizado não supervisionado). Foram utilizados três algoritmos: K-Means (baseado em centróides), DBSCAN e OPTICS (ambos baseados em densidade). O objetivo central foi avaliar o desempenho de cada algoritmo na segmentação de perfis de clientes com base no coeficiente de Silhouette.

As bases de dados (oriundas do sítio *Kaggle*) utilizadas totalizam quatro: “pagamento dos pedidos”, “*reviews* dos clientes por pedido”, “pedidos”, “clientes” e “items pedidos”. A base de dados fornecido pela Olist abrange mais arquivos, porém, para este estudo de caso foram escolhidas cinco de todas as nove.

1 Acadêmico do Curso de Big Data e Ciência Analítica em 2025; E-mail: 5194418@aluno.uniasselvi.com.br

2 Tutor Externo do Curso de Big Data e Ciência Analítica em 2025; E-mail: guilherme.santos@regente.uniasselvi.com.br

Inicialmente, foi realizada uma análise exploratória dos dados com visualização dos tipos de dados, mesclagem das cinco bases de dados, criação de variáveis auxiliares, detecção de outliers, preenchimento de valores ausentes e normalização dos atributos (com o “StandardScaler”) para melhorar o desempenho dos algoritmos, visto que a distância entre pontos afeta tanto algoritmos baseados em centróides quanto em densidade. As variáveis mais relevantes para o agrupamento foram selecionadas com o objetivo de segmentar os clientes em razão do perfil de consumo, a opinião deles em relação à empresa e, por último, diminuir a dimensionalidade da base de dados. A seguir, foram aplicados os algoritmos de clustering mantendo os hiperparâmetros parecidos como ato de experimentação e tanto a escolha do número de clusters e a eficiência de cada um foi avaliado conforme a minimização da inércia (este, também chamado de erro quadrático é utilizado no algoritmo K-Means) e a maximização do coeficiente de Silhouette (para os algoritmos baseados em densidade) e a coerência visual dos agrupamentos.

O uso dessas técnicas proporcionou uma compreensão mais detalhada sobre os perfis de clientes existentes e ofereceu diretrizes para estratégias de marketing direcionadas.

CONSIDERAÇÕES CRÍTICAS E CRIATIVAS

Python foi a linguagem de programação utilizada neste estudo de caso. Diferente do trabalho proposto no semestre anterior, não é pedido uma análise de dados detalhada na trilha de aprendizagem, logo, o foco é a modelagem, a comparação e extração de resultados.

É um conjunto de dados que reflete operações de comércio eletrônico no Brasil, hospedado na plataforma Kaggle. Contém diversas tabelas inter-relacionadas (como pedidos, produtos, clientes, pagamentos, revisões e geolocalização) que possui informações de 100 mil pedidos de 2016 a 2018 feitos em vários locais no Brasil, onde seus recursos permitem visualizar um pedido em várias dimensões.

A seguinte problemática fictícia que é objeto de resolução neste estudo de caso: “Foi solicitado pela empresa uma segmentação de perfil de clientes com base na quantidade de compras feitas, o objetivo é direcionar o time de marketing e vendas para atrair clientes e não perder os que não compram há um tempo considerável.”

A primeira parte se trata da forma do dataset. As bases de dados originais foram mescladas e resultou em um *dataset* de 117.329 linhas e 28 colunas.

Figura 01 – Tamanho do dataset e tipos de colunas

```
# Exibindo a base agrupada
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117329 entries, 0 to 117328
Data columns (total 28 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   customer_id                    117329 non-null  object
 1   customer_unique_id             117329 non-null  object
 2   customer_zip_code_prefix       117329 non-null  int64
 3   customer_city                  117329 non-null  object
 4   customer_state                 117329 non-null  object
 5   order_id                       117329 non-null  object
 6   order_status                   117329 non-null  object
 7   order_purchase_timestamp       117329 non-null  object
 8   order_approved_at              117314 non-null  object
 9   order_delivered_carrier_date   116094 non-null  object
 10  order_delivered_customer_date  114858 non-null  object
 11  order_estimated_delivery_date  117329 non-null  object
 12  review_id                      117329 non-null  object
 13  review_score                   117329 non-null  int64
 14  review_comment_title           13892 non-null   object
 15  review_comment_message         49679 non-null   object
 16  review_creation_date           117329 non-null  object
 17  review_answer_timestamp        117329 non-null  object
 18  payment_sequential             117329 non-null  int64
 19  payment_type                   117329 non-null  object
 20  payment_installments           117329 non-null  int64
 21  payment_value                  117329 non-null  float64
 22  order_item_id                  117329 non-null  int64
 23  product_id                     117329 non-null  object
 24  seller_id                      117329 non-null  object
 25  shipping_limit_date            117329 non-null  object
 26  price                          117329 non-null  float64
 27  freight_value                  117329 non-null  float64
dtypes: float64(3), int64(5), object(20)
memory usage: 25.1+ MB
```

Fonte: Autor

A segunda parte envolve a criação de variáveis (colunas) auxiliares para enriquecer os modelos de *clustering*. Seis f*eatures* foram criadas e mescladas ao *dataset* original: total de compras por cliente, tempo médio de entrega dos pedidos, valor médio do frete, total gasto por cliente, média da avaliação do cliente para os pedidos (review score) e quantas parcelas em média o cliente fez.

Na análise exploratória de dados foram removidas as colunas dispensáveis ao estudo de caso, as *features* que não se alinham ao escopo do projeto e podem não enriquecer os modelos de *machine learning* utilizados.

Figura 02 – Colunas removidas

```
colunas_para_remover = [
    'customer_id', 'order_id', 'customer_zip_code_prefix', 'order_status',
    'order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date',
    'order_delivered_customer_date', 'order_estimated_delivery_date', 'review_id',
    'review_score', 'review_comment_title', 'review_comment_message',
    'review_creation_date', 'review_answer_timestamp', 'payment_sequential',
    'payment_type', 'payment_installments', 'payment_value', 'order_item_id',
    'product_id', 'seller_id', 'shipping_limit_date', 'price', 'freight_value',
    'customer_city', 'customer_state'
]
```

Fonte: Autor

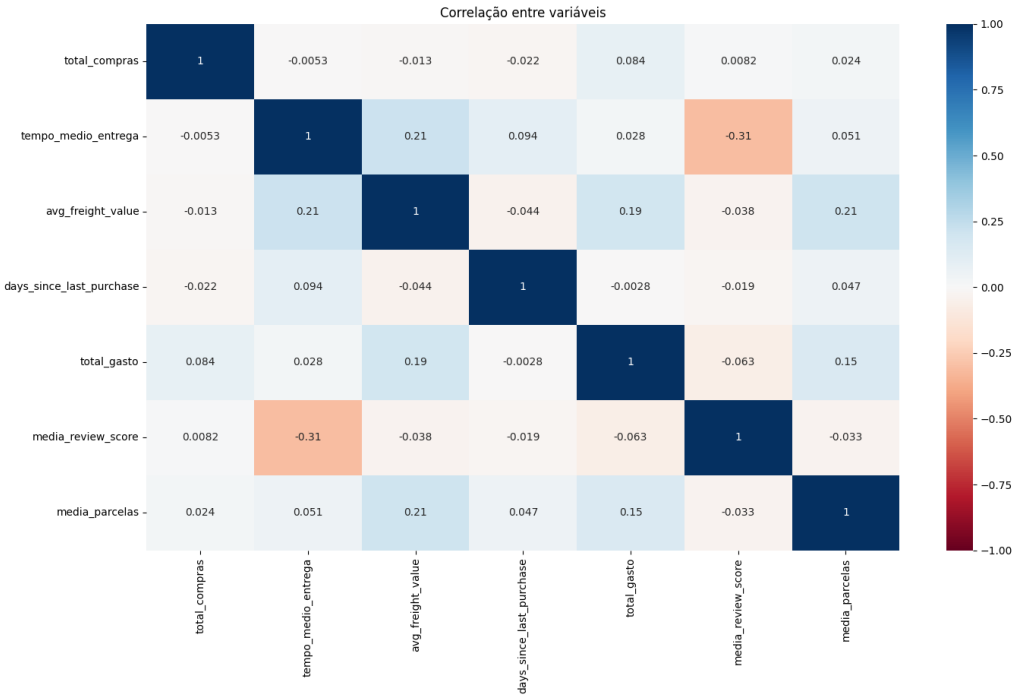
A terceira parte abrange a identificação, preenchimento ou remoção de valores nulos e o tratamento de valores duplicados. A única variável com 2.325 valores ausentes era *tempo_medio_entrega, estes espaços foram* preenchidos com a

mediana, enquanto as 22.609 duplicatas foram removidas mediante o método *.drop_duplicates()*.

A quarta parte aborda a normalização do *dataset e a* menção da utilização do *Principal Component Analysis*. A normalização foi executada com *StandardScaler* da biblioteca *sklearn.preprocessing* que coloca os dados em uma distribuição normal de média zero e desvio-padrão 1.

Para experimentação e verificação de desempenho, alguns dos modelos foram utilizados na base de dados com a técnica de redução de dimensionalidade chamada Análise de Componentes Principais (*PCA* em inglês). No entanto, *Hair et al (2009)* sugere o uso do *PCA* quando a maior parte das correlações assume valores absolutos maiores que 0,3, não é o caso deste conjunto de dados. Por se tratar de um estudo de caso fictício, decidi por experimentar a aplicação do *PCA* mesmo que a correlação entre as *features* seja menor que 0.3.

Figura 03 – Correlação de Pearson entre variáveis

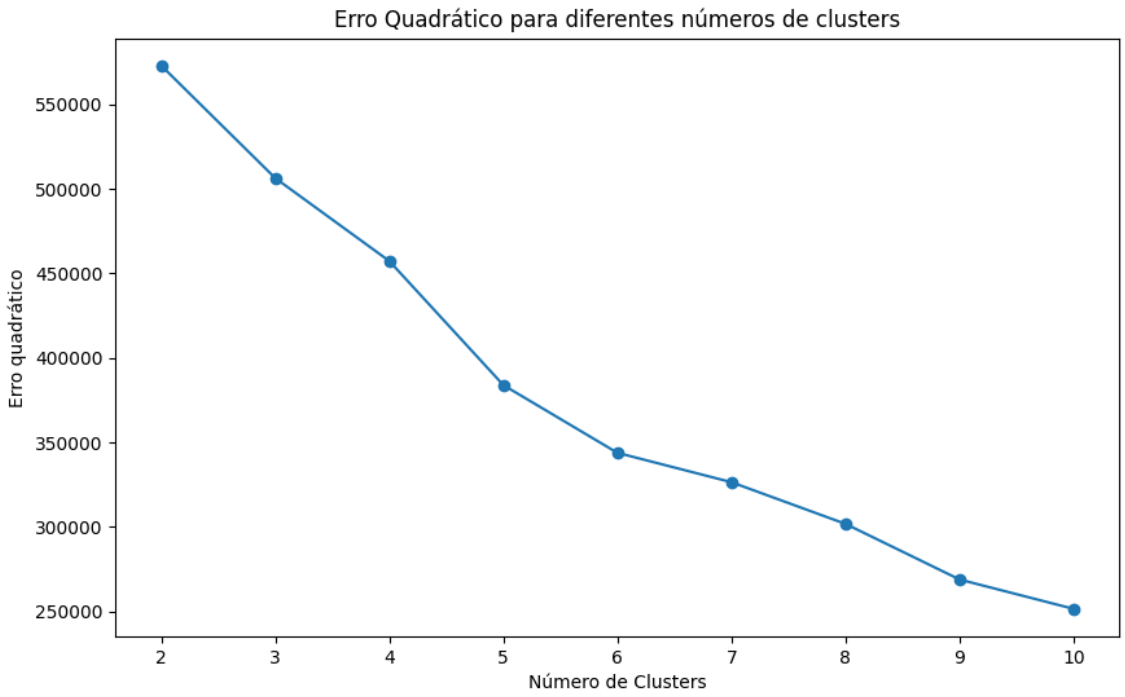


Fonte: Autor

O *PCA* é uma técnica que divide a base de dados em componentes principais, estes componentes maximizam a variância e diminuem a correlação dos dados, assim, possibilitando uma melhor divisão de *clusters* enquanto mantém a essência de informação dos dados.

A quinta parte envolve a aplicação dos algoritmos de agrupamento não supervisionado. Foram utilizados três algoritmos: K-Means, DBSCAN e OPTICS. O K-Means foi o primeiro a ser testado e será usado como referência para os outros.

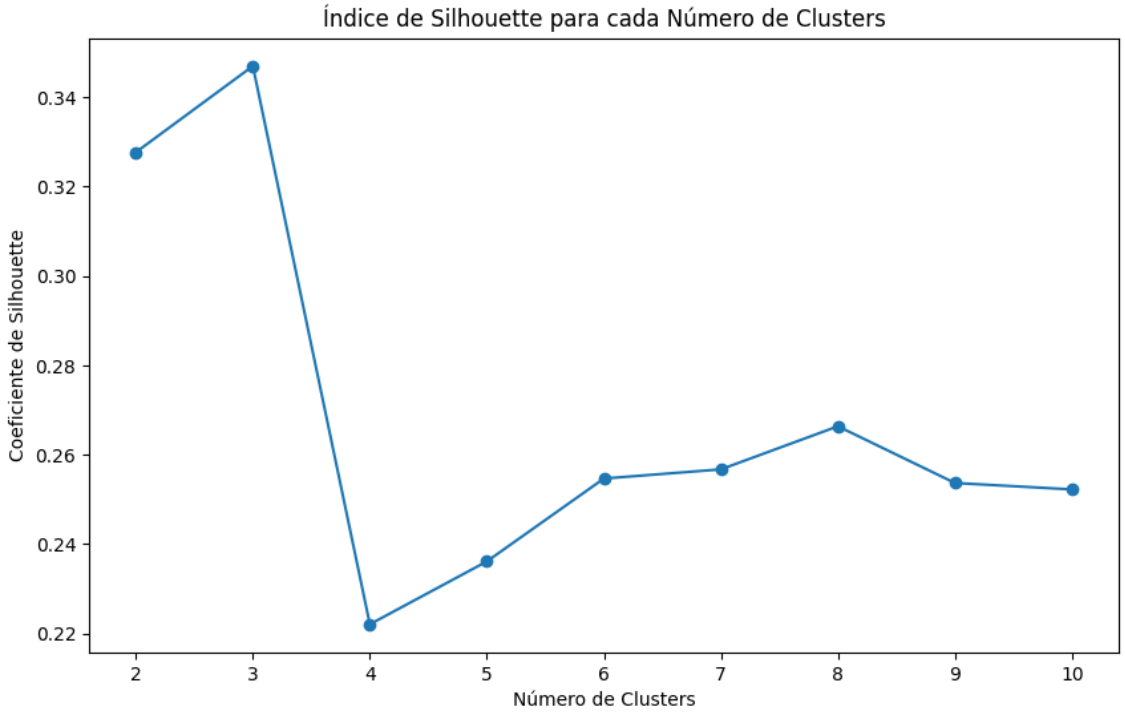
Figura 04 – Erro Quadrático (inércia) no K-Means sem *PCA*



Fonte: Autor

A inércia (ou erro quadrático) mede a distância entre os pontos de dados em um *cluster.* O método do "cotovelo" para decisão do número de clusters é bem subjetivo, podendo duas pessoas diferentes identificar pontos distintos da curva como ideal. Dito isto, é possível aplicar outro coeficiente para decidir o número de clusters chamado *Silhouette Score*.

Figura 05 – Coeficiente de Silhouette no K-Means sem *PCA*

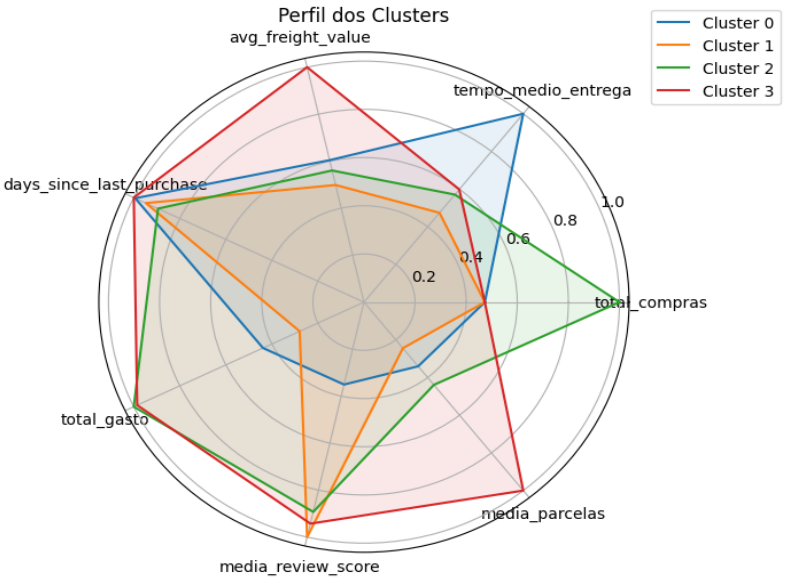


Fonte: Autor

Aproximadamente 450 mil de inércia e 0.22 do Índice de *Silhouette* para 4 *clusters*. Como o objetivo do estudo de caso é segmentar perfis, opto por quatro *clusters*. O correto seria o maior índice de *Silhouette* e a menor inércia (erro quadrático).

Executei uma análise univariada separando os clientes por *clusters*, porém, uma análise multivariada permitiu a visualização de todos os perfis em um único gráfico e gerou uma visão melhor, confirmando a análise univariada.

Figura 06 – G*ráfico de radar* do perfil de clientes nos *clusters* do *K-Means*



Fonte: Autor

Existe uma métrica chamada RFM que analisa os clientes em somente três aspectos: recência (R), frequência (F) e monetariedade (M) todos de zero à cinco ou zero à dez. Essa métrica visa segmentar os clientes ordenando e classificando como melhores os que obtiverem menor recência, maior frequência e maior monetariedade, gerando um nota geral para cada um.

Seria necessário ordenar os clientes caso utilizasse puramente esta métrica, contudo, permite uma perspectiva muito pobre em uma base de dados tão extensa como esta. É interessante utilizar a ótica de classificação dos clientes pelo RFM nos *clusters* que se obteve. Fato é que o gráfico de radar está normalizado com os valores entre zero e um, assim é possível identificar o que cada *cluster* agrupou em uma escala de zero à dez sem ordenar os clientes. A recência é representada por *days_since_last_purchase*, frequência por *total_compras* e monetariedade por *total_gasto*.

Uma observação é que caso fosse utilizado o RFM antes da aplicação dos modelos não-supervisionados como colunas da base de dados (o que é comumente feito), reforçaria um viés na segmentação de clientes usando a monetização, frequência e recência, sendo que estes três já existem como *features* (*total_gasto*, *total_compras* e *days_since_last_purchase*). Prefiro avaliar estes grupos com as *features* existentes e analisar os *clusters* resultantes com a perspectiva do RFM. Não há uma convenção sobre a ordem de execução.

De acordo com o K-Means sem PCA: o *cluster* 0 é um grupo de novos clientes ou clientes de baixo valor. Possuem frequência baixa, valor monetário baixo, parcelamento baixo, *review* alto e recência boa (acabaram de comprar).

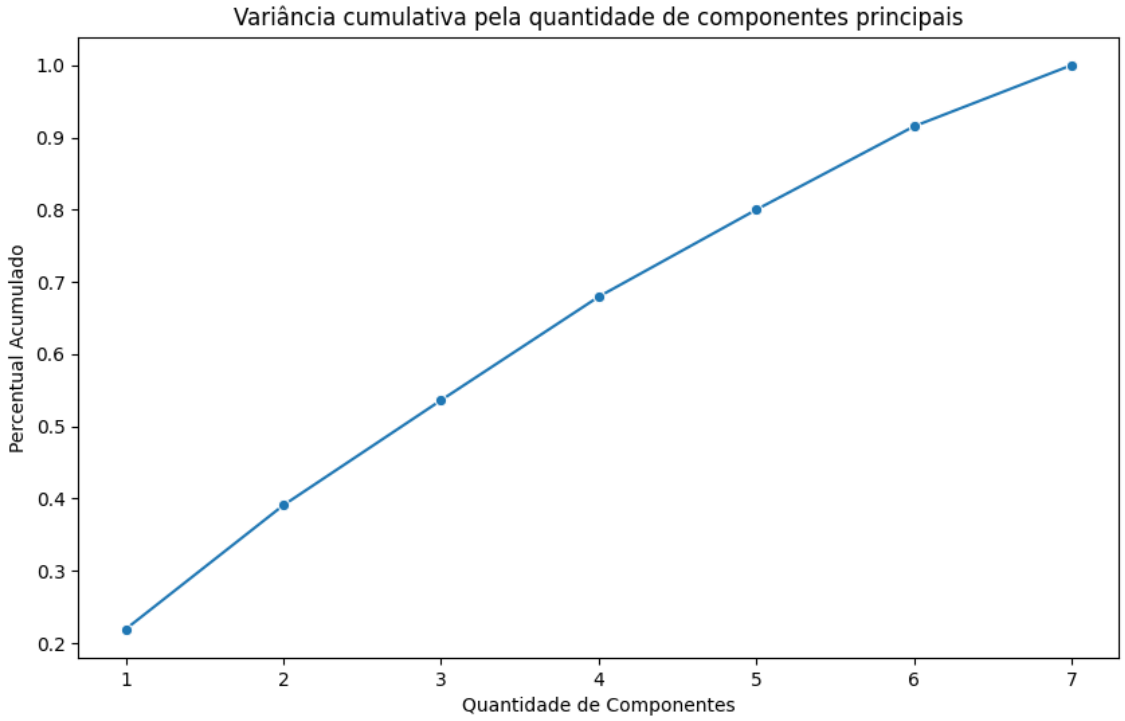
O *cluster* 1 é um grupo de clientes promissores. *Review score* alto, valor monetário bom e frete alto, parcelamento alto, recência boa e frequência baixa (total_compras baixo).

O *cluster* 2 é o grupo de melhores clientes (campeões do RFM). Recência baixa (sinal ruim de que compraram há bastante tempo) frequência alta e alto valor monetário.

O *cluster* 3 é uma parcela de clientes em risco ou insatisfeitos. Recência muito ruim, valor monetário baixo, *review* score muito baixo, frequência baixa, tempo médio de entrega mais alto.

A sexta parte aborda a decisão do número de componentes principais.

Figura 07 – Variância cumulativa pela quantidade de componentes principais



Fonte: Autor

Tabela 1. Variância por componente principal

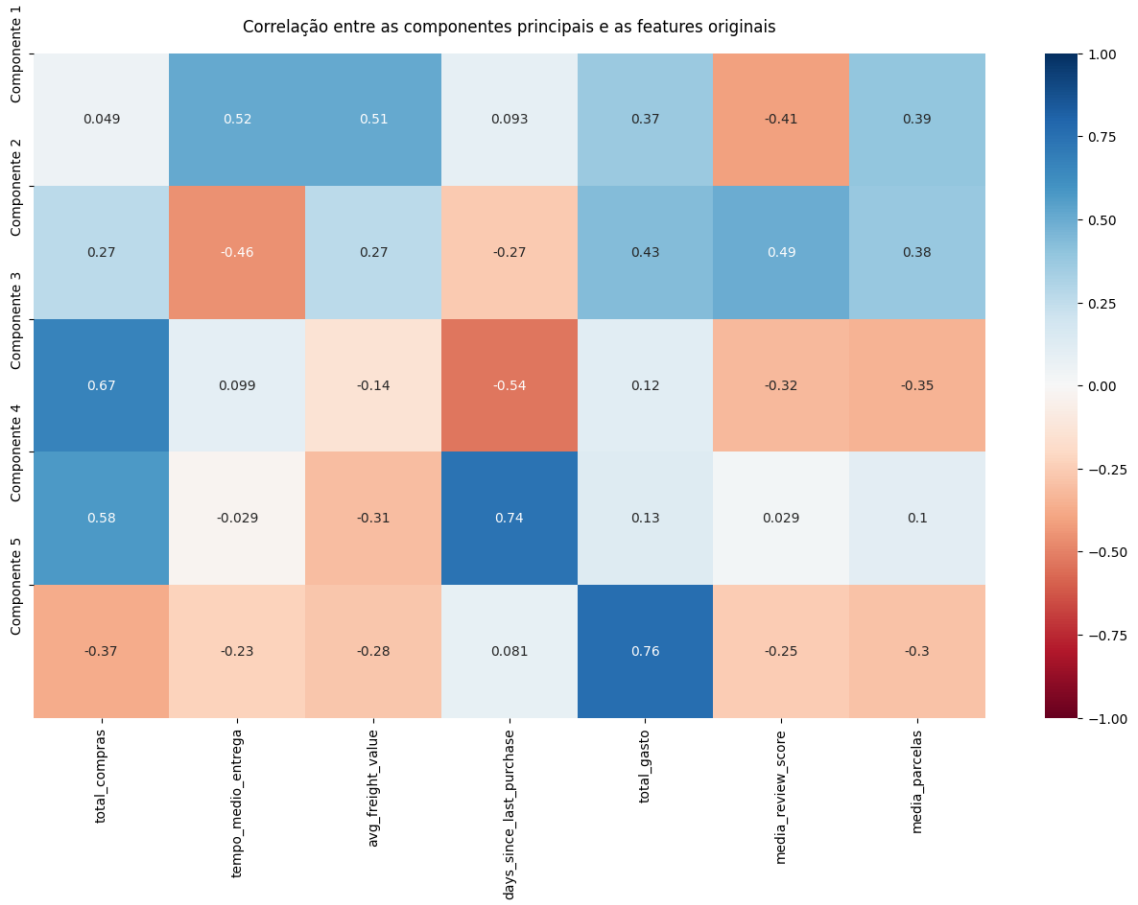
Componentes Principais	Variancia_Explicada	Variancia_Cumulativa
PC1	**0.219081**	0.219081
PC2	**0.171546**	0.390627
PC3	**0.145396**	0.536023
PC4	**0.143310**	0.679333
PC5	**0.120578**	**0.799911**
PC6	0.115362	0.915273
PC7	0.084727	1.000000

Fonte: Autor

Decido por manter 80% da variância com cinco componentes principais e também experimentar com dois componentes principais para avaliar a segmentação

em gráficos 2D. É possível visualizar qual componente possui mais informação de determinada *feature* original analisando o método pca.components_ por coluna ou simplesmente exibir a correlação de Pearson entre os componentes principais e as *features* originais. Por exemplo: O componente principal 1 (PC1) possui 4% de informação de *total_compras,* enquanto *PC3* possui 66% da variância original da mesma *feature*.

Figura 08 – Mapa de calor entre *PCA* e *features* originais



Fávero et al. (2009) destaca que nem sempre é simples nomear um fator, uma vez que bases com grande quantidade de *features* podem gerar poucos fatores, dificultando captar a essência de todas as variáveis.

Em um case real, deve-se nomear cada *feature* componente principal de acordo com sua essência, neste caso a essência de cada um corresponde à alta correlação com as variáveis originais. Onde, por exemplo, o componente 5 absorveu maior parte do valor despendido pelo cliente na empresa, poderia ser renomeado para *total_gasto*. Quanto maior a quantidade de features com correlação positiva, maior a dificuldade de nomear os componentes principais em razão da natureza do PCA capturar a essência do *dataset* de diferentes formas.

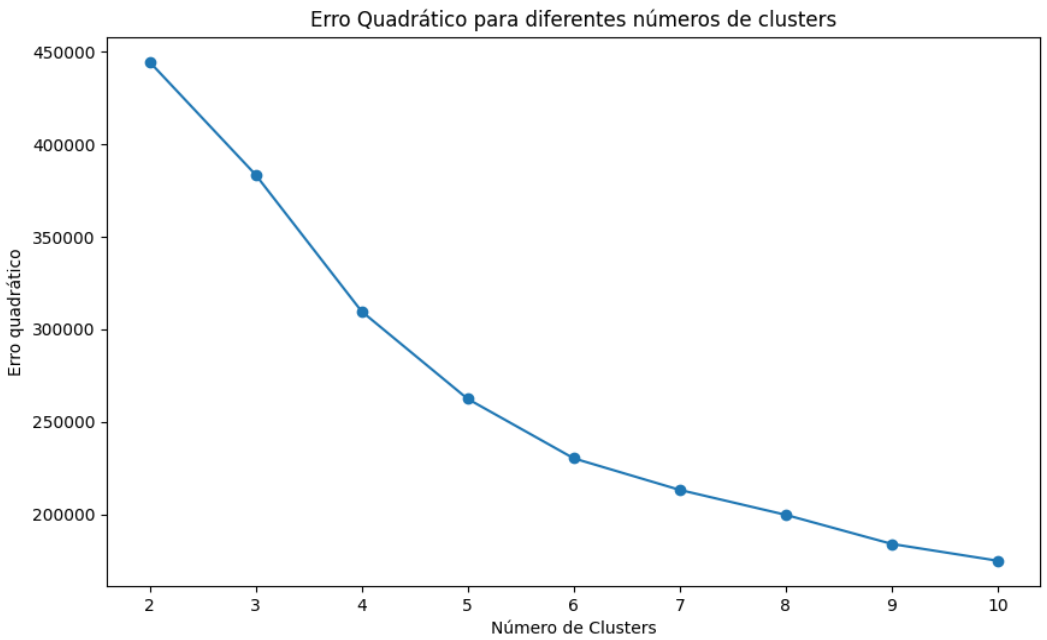
Análise do mapa de calor para redução de dimensionalidade com *PCA* de 7 para 2 dimensões:

- **Componente 1: Tempo médio de entrega e valor médio do frete**
  - Eixo dos dados que aborda o tempo médio de entrega e o valor médio do frete, guardou com pouca essência o total gasto e a quantidade de parcelas.
  - *tempo_medio_entrega*: 0.52 Correlação moderada positiva
  - *avg_freight_value*: 0.51 Correlação moderada positiva
  - *total_gasto*: 0.37 Correlação fraca positiva
  - *media_parcelas*: 0.39 Correlação fraca positiva
- **Componente 2: Média de notas das *reviews* e dias desde a última compra**
  - Parte dos dados que aborda a média das notas de *review*, dias desde a última compra e a quantidade de parcelas na mesma proporção que PC1. *total_compras* não é bem representado por este eixo devido à baixa correlação.
  - *media_review_score*: 0.49 Correlação moderada positiva
  - *avg_freight_value*: 0.27 Correlação fraca positiva
  - *days_since_last_purchase*: 0.43 Correlação moderada positiva
  - *media_parcelas*: 0.38 Correlação fraca positiva
  - *total_compras*: 0.27 Correlação fraca positiva

Observação: no *jupyter notebook* é possível visualizar os nomes dos outros componentes principais (PC3, PC4 e PC5) que não são relevantes no resultado final deste objeto de estudo.

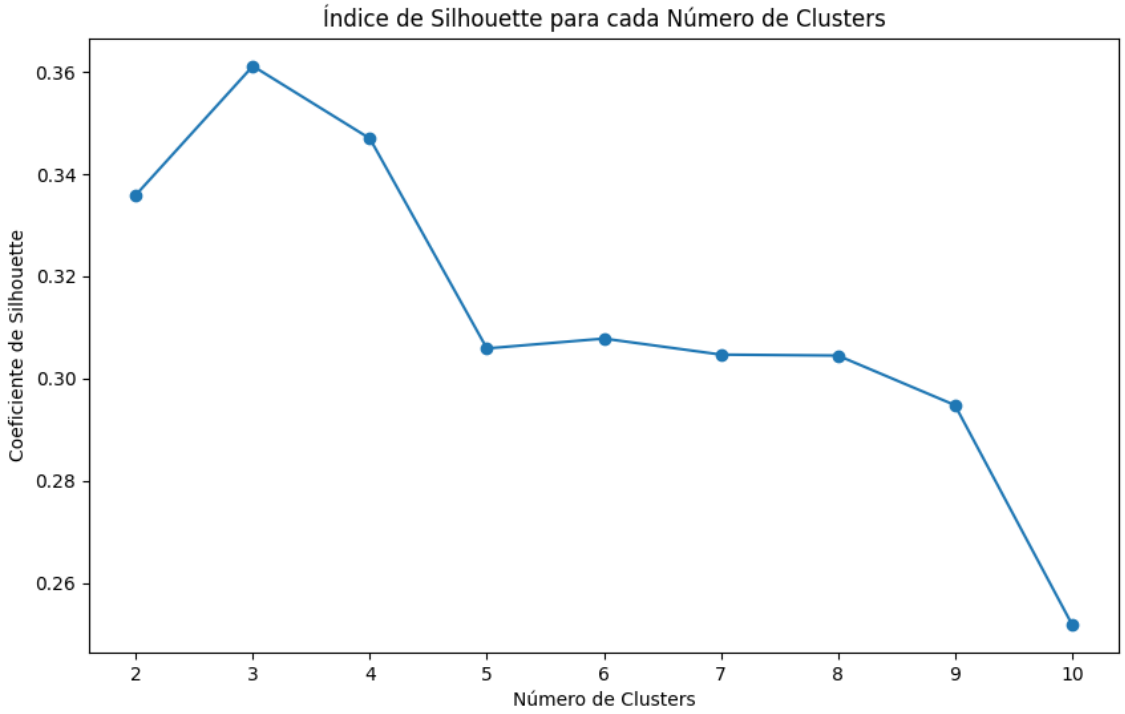
A sétima parte aborda a re-aplicação do *K-Means* com *PCA* e comparação das métricas.

Figura 09 – Erro Quadrático no K-Means com PCA



Fonte: Autor

Figura 10 – Coeficiente de *Silhouette* no *K-Means* com *PCA*

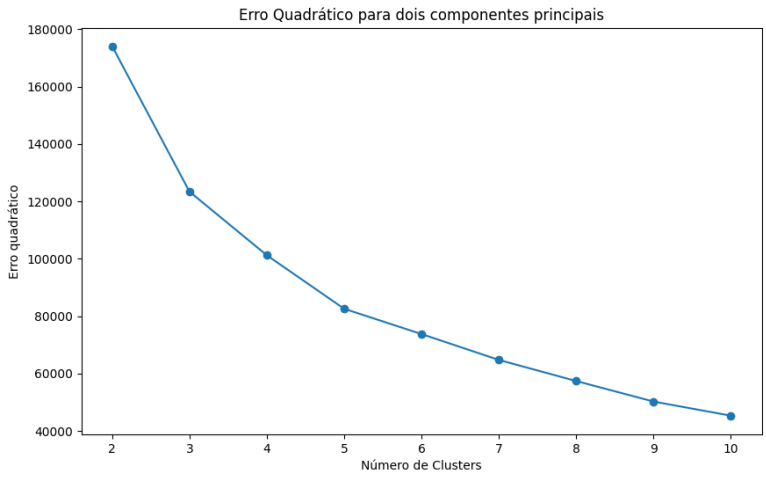


Fonte: Autor

É possível observar que para quatro *clusters* o erro quadrático diminuiu em mais de 100 mil unidades comparado com o anterior. Indicando uma melhora na coerência da segmentação. O índice de *Silhouette* apresentou uma melhora de um pouco mais de um décimo (0,1) com quatro *clusters* se comparado o anterior, indicando uma melhor coesão intra-*cluster* e separação inter-*cluster* dos agrupamentos.

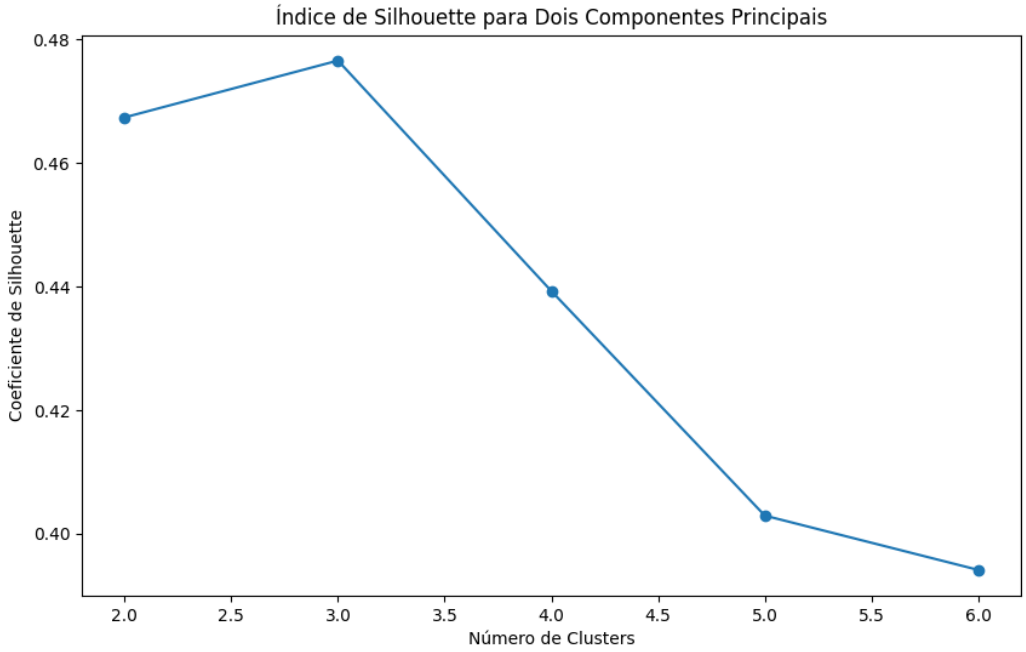
A oitava parte se trata de analisar as métricas do K-Means sobre dois componentes principais.

Figura 11 – Inércia no K-Means para dois componentes principais



Fonte: Autor

Figura 12 – *Silhouette score* no *K-Means* para dois componentes principais

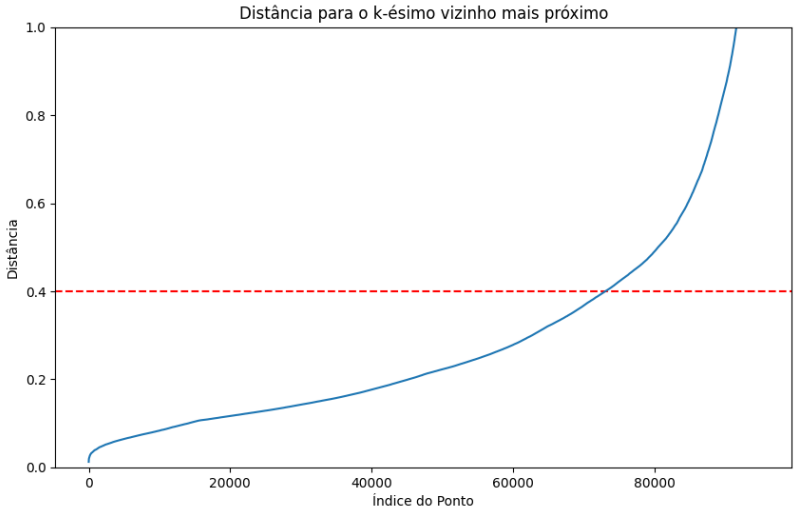


Fonte: Autor

A inércia diminuiu (como esperado) de aproximadamente 300.000 para 100.000 tratando-se de 4 *clusters*. O coeficiente de *Silhouette* aumentou de 0.34 para aproximadamente 0.44 (para 4 *clusters*) confirmando que, às vezes, menor variância também significa *clusters* mais bem definidos. Foi verificado que o desempenho do *K-Means* com cinco componentes principais foi inferior quando comparado ao mesmo modelo com dois componentes principais.

A nona parte descreve a aplicação do algoritmo DBSCAN com *PCA, a* decisão do valor do parâmetro épsilon e compara com K-means. É possível utilizar o algoritmo Nearest Neighbors para visualizar a distância entre pontos de um dataset e identificar o “cotovelo” para definir o parâmetro épsilon.

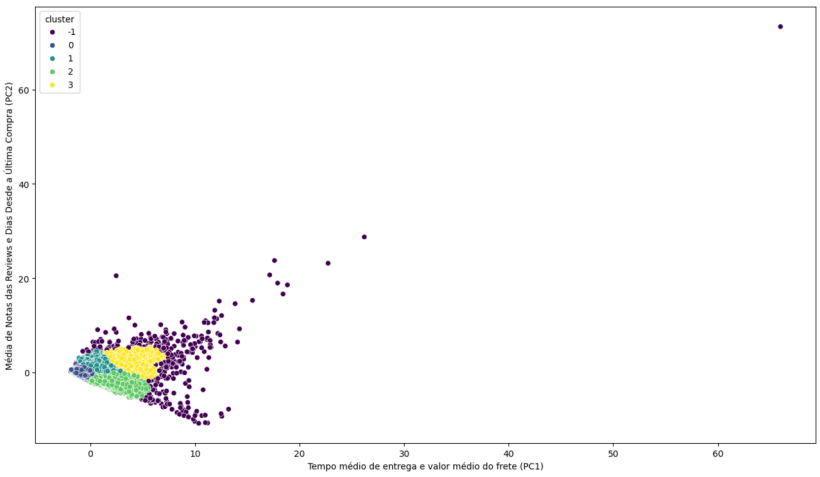
Figura 13 – Distância para o k-ésimo vizinho mais próximo



Fonte: Autor

O épsilon define a distância máxima entre dois pontos de dados para que um seja considerado como vizinho de outro. A região em que a distância não cresce significativamente é no 0,4. Antes do "cotovelo" as distâncias são maiores. Após o "cotovelo" são áreas com menor densidade e com maior distância, isso resultaria no agrupamento de dados ruidosos e dados considerados como *outliers*.

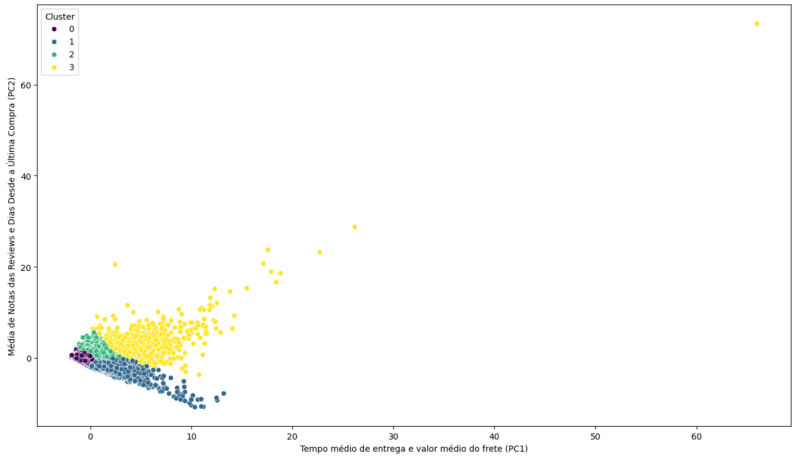
Figura 14 – Clusters do DBSCAN com dois componentes principais



Fonte: Autor

DBSCAN identificou 4 clusters, sendo o "-1" (cor roxa) um aglomerado de outliers que somam 302 instâncias. Está muito melhor agrupado quando comparado com o K-Means utilizando dois componentes principais.

Figura 15 – Clusters do K-Means com dois componentes principais



Fonte: Autor

É possível, ainda comparar o Índice de Silhouette do K-Means com o DBSCAN já que algoritmos não-supervisionados não atuam em bases de dados com rótulos.

Figura 16 – Código exibindo *Silhouette score* do DBSCAN

```
# Índice de Silhouette, identificando se o objeto está bem alocado no cluster.
silhouette = list()

dbscan_ss = DBSCAN(eps=0.4, min_samples=8).fit(df_pca2)
silhouettescore = silhouette_score(df_pca2, clustering.labels_)

silhouette.append(silhouettescore)

print(silhouette)
```

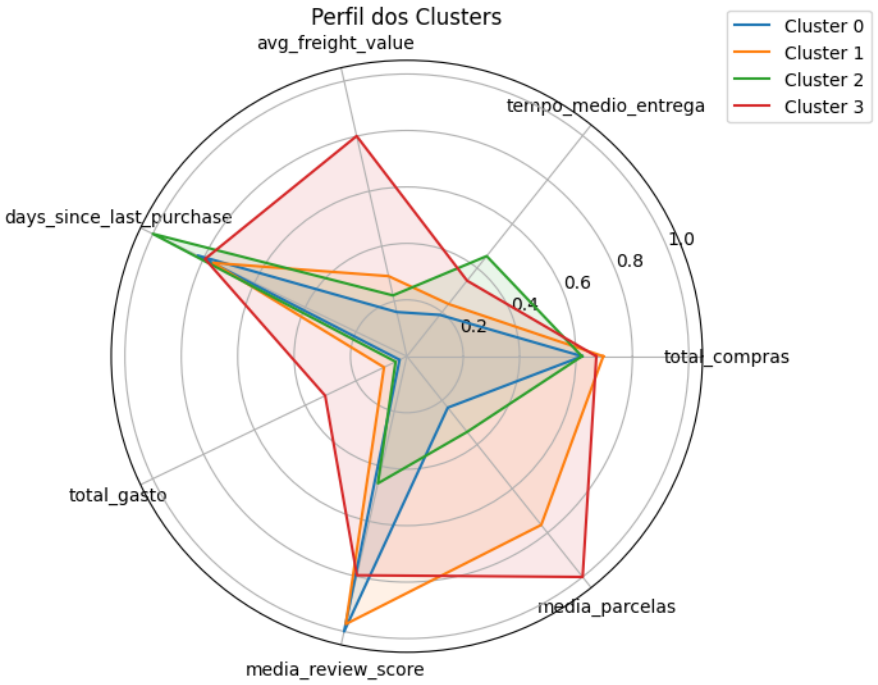
```
[0.5327537512265761]
```

Fonte: Autor

O coeficiente de Silhouette para o DBSCAN foi de 0.53, melhor que o KMeans (com dois componentes principais) cujo coeficiente resultou em 0.44, ambos para 4 clusters. Os outliers podem apresentar informações interessantes em situações reais, mas neste case fictício prefiro ignorá-los. O DBSCAN encontrou diferentes densidades de forma coesa.

A décima parte faz uma análise do perfil de clientes (com a ótica do RFM novamente) por clusters com a segmentação de clientes do DBSCAN.

Figura 17 - G*ráfico de radar* do perfil de clientes nos *clusters* do DBSCAN



Fonte: Autor

Figura 18 – Perfil dos clusters (DBSCAN) com base na métrica RFM (célula *markdown* do *jupyter notebook*)

Cluster	Classificação RFM	Destaques	Ação recomendada
0	Fiéis de baixo ticket	Avaliações excelentes, mas pouco valor monetário	Campanhas personalizadas
1	Promissores / Leais	Engajamento alto, avaliações excelentes, ticket intermediário	Fidelização, antecipação de ofertas
2	Dormindo / Inativos	Alta recência, baixo valor e review péssimo	Reativação, diagnóstico de abandono
3	Clientes em rísco	Logística cara, gasto médio, avaliações medianas	Descontos no frete e retenção estratégica

Fonte: Autor

A décima parte abrange a aplicação do algoritmo OPTICS com e sem *PCA* utilizando os mesmos valores de épsilon e *min_samples*.

Figura 19 – OPTICS sem *PCA*

```
# OPTICS sem PCA
# define os parâmetros do modelo OPTICS
optics_model = OPTICS(eps=0.4, min_samples=8)

# determina um cluster para cada ponto
optics_result = optics_model.fit_predict(scaled_features)
```

```
# OPTICS sem PCA
# acrescentando os rótulos
df_optics = df.copy()
df_optics['cluster'] = optics_result

#df_optics.head()
```

```
Outputs are collapsed ···
```

```
df_optics['cluster'].value_counts().sort_index()
```

```
cluster
-1       81065
 0           8
 1           9
 2          10
 3          14
          ...
 1032        8
 1033        9
 1034       19
 1035       28
 1036        9
Name: count, Length: 1038, dtype: int64
```

Fonte: Autor

Optics sem PCA identificou 81 mil instâncias como *outliers* e também identificou mil *clusters*. O resultado ao reproduzir o mesmo algoritmo na base de dados com dois componentes principais foi a identificação de 2846 clusters e 56 mil instâncias classificadas como outliers (o dataset engloba 100 mil instâncias). Optei por não exibir gráficos e nem obter o índice de *silhouette* do OPTICS em razão da quantidade de agrupamentos não ser interpretável. Vale ressaltar que para dados bidimensionais, é interessante usar o valor padrão do DBSCAN de MinPts = 4 (Ester

et al., 1996). Se os dados tiverem mais de 2 dimensões, escolher MinPts = 2**dim*, onde *dim* é igual as dimensões do seu conjunto de dados (Sander et al., 1998), onde MinPts nos algoritmos baseados em densidade se chamam min_samples. É uma das experimentações a ser feitas em um case real para tentar corrigir o número exagerado de clusters identificado pelo OPTICS e talvez até melhorar o *Silhouette score* dos outros algoritmos utilizados.

A décima primeira parte envolve a conclusão. O algoritmo que se saiu melhor foi o DBSCAN com dois componentes principais, de acordo com o Índice de Silhouette. Os próximos passos em um cenário real seriam:

- Concatenar a base de dados para ser possível identificar cada cliente por um ID;
- Separar os clientes em grupos: os que receberão as campanhas personalizadas e os que não receberão;
- Traduzir as informações obtidas no estudo de caso de uma forma interpretável para os times de marketing e vendas;
- Após a implementação das estratégias de campanhas personalizadas pelos times supracitados, observar os resultados, comparar com períodos anteriores, comparar também com o grupo que não recebeu as campanhas e identificar pontos de melhoria visando maximizar o lucro; e
- Uma opção ainda, seria dividir os clientes em mais grupos, utilizar os KMeans e DBSCAN para fazer diferentes campanhas e identificar qual algoritmo representou a realidade na prática, isto é, nos grupos que receberam e não receberam o marketing direcionado. Uma espécie de teste *A/B* e avaliação de resultados.

REFLEXÕES FINAIS

Concluir este objeto de estudo acrescentou experiência prática do que um empreendedor exigiria de um cientista de dados: alinhamento, ações estratégicas baseadas em dados e conversão de resultados em dinheiro. É importante ressaltar que este projeto foi desenvolvido com o propósito de resolver um problema fictício de negócio.

Destaca-se que a segmentação de dados através de algoritmos não supervisionados é uma ferramenta poderosa quando combinada com o conhecimento do contexto de negócio. O projeto evidencia que técnicas bem aplicadas podem transformar grandes volumes de dados em conhecimento estratégico, contribuindo para decisões empresariais mais embasadas e eficazes.

Uma visualização mais detalhada do processo de análise, código, modelagem e resultados pode ser verificado no *github* disponível a seguir: https://github.com/Gbibiano/clustering-e_commerce/blob/main/e_commerce.ipynb.

REFERÊNCIAS

Base de dados utilizada: “Brazilian E-Commerce Public Dataset by Olist”. Disponível em https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce.

HAIR, Joseph F. et al. Análise multivariada de dados. Bookman editora, 2009.

Miglautsch, JR (2000). "Reflexões sobre a pontuação RFM." Journal of Database Marketing & Customer Strategy Management, 8(1), 67-72.

FÁVERO, Luiz Paulo Lopes et al. Análise de dados: modelagem multivariada para tomada de decisões. 2009.

Patel, A (2019). Hands-On Unsupervised Learning Using Python. USA: O’Reilly Media, Inc.. 166.

Sander, J., Ester, M., Kriegel, HP. et al. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. Data Mining and Knowledge Discovery 2, 169–194 (1998).