UNIVERSIDADE FEDERAL DO PARANÁ

GABRIEL ALHER DE CARVALHO

ALGORITMOS DE CLUSTERIZAÇÃO E O MODELO RFV (RECÊNCIA, FREQUÊNCIA E VALOR) APLICADOS EM DADOS DE *E-COMMERCE*

CURITIBA

2020

GABRIEL ALHER DE CARVALHO

ALGORITMOS DE CLUSTERIZAÇÃO E O MODELO RFV (RECÊNCIA, FREQUÊNCIA E VALOR) APLICADOS EM DADOS DE *E-COMMERCE*

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia de Produção, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia de Produção.

Orientadora: Profª. Drª. Mariana Kleina

CURITIBA

2020

RESUMO

O mercado consumidor está mais exigente e mudando cada vez mais rapidamente. O *e-commerce*, canal sem relevância há alguns anos, hoje é bastante lucrativo e cheio de oportunidades. Por isso, entender o perfil de seu consumidor para traçar estratégias mais assertivas serve como vantagem competitiva para qualquer empresa frente à concorrência. Neste trabalho foram aplicados dois algoritmos de clusterização, o *k-means* e o AGNES, para agrupar clientes com perfis semelhantes de um *e-commerce*, baseado nas três métricas que compõem o modelo RFV: recência, frequência e valor. Foram analisados os dados de pedidos realizados entre setembro de 2016 a setembro de 2018 no *marketplace Olist*. Para o conjunto de dados selecionado, o método *k-means* se mostrou mais eficiente que o método AGNES, pois seu Coeficiente de Silhueta foi maior em 70% das amostras analisadas, além da eficiência computacional do algoritmo. Os clientes desse *e-commerce* foram agrupados em quatro *clusters*, para os quais foram sugeridas ações de retenção e rentabilização.

Palavras-chave: Clusterização. Modelo RFV. *K-means*. AGNES. *e-commerce*.

ABSTRACT

The consumer market is more demanding and changing more and more rapidly. E-commerce, a channel with no relevance for some years, today is very profitable and full of opportunities. Therefore, knowing the consumer's profile to create assertive strategies is a competitive advantage for any company to face up the competition. At this study there were two clustering algorithms applied, K-means and AGNES, to unite clients from e-commerce with similar profiles, based on three metrics which are combined into the RFV model: recency, frequency and value. The data for orders placed between September 2016 and September 2018 on the Olist marketplace were analyzed. For the selected data set, the k-means method indicates more efficiency than AGNES method, due to its Silhouette Coefficient being higher on seventy percent of the analyzed samples, besides the efficient computational algorithm. Clients from this e-commerce were grouped into four clusters, for which were suggested retention and profitable actions.

Keywords: Clustering. RFV model. K-means. AGNES. E-commerce.

SUMÁRIO

1 INTRODUÇÃO

"*E-commerce* cresce além do esperado, aponta *Ebit Nielsen*". A notícia, veiculada pelo jornal Meio & Mensagem em fevereiro de 2020, indica avanço de 16,3% do comércio eletrônico em 2019, quando comparado com o ano anterior. Estima-se que, no Brasil, foram realizados 148,9 milhões de pedidos pela *internet*, totalizando um valor faturado de R$ 61,9 bilhões (MEIO E MENSAGEM, 2020).

O *e-commerce*, que já apresentava tendências de crescimento desde o início deste ano, foi bastante impulsionado pela pandemia da *covid-19*. Segundo a notícia da revista Exame, em virtude do vírus, as compras *online* já aumentaram 40% no Brasil. De acordo com a revista Época, o comércio eletrônico em 2020 foi 48,3% maior do que o realizado no mesmo período de 2019 (dados de março e abril).

A migração para o comércio *online* é fomentada também pelo acesso cada vez maior à *internet*, computadores e *smartphones*. Em uma pesquisa realizada em 2015 pela Fecomercio, 75% dos entrevistados informaram que utilizam a *internet* e as redes sociais para obter informações para aquisição de produtos e serviços. Atualmente, cinco anos depois, o movimento de grandes *players* do varejo brasileiro em resposta a esse novo público é perceptível: aplicativos com promoções exclusivas, *e-mail marketings* cada vez mais assertivos e *sites* com diversos atributos que melhoram a experiência do consumidor.

Por fim, a integração dos canais virtuais com a loja física, integração esta conhecida por *omnichannel*, vem ganhando gradativamente mais espaço. Por exemplo, já é comum que compras sejam feitas pela *internet*, mas a retirada dos produtos seja feita na loja física. A '*multicanalidade*', como essa integração pode ser traduzida, também figura como uma tendência quando se fala de *e-commerce.*

Na era do *Big Data*, a quantidade de informações e técnicas disponíveis para gerar conhecimento sobre os clientes de *e-commerce* é bastante ampla. Ao contrário de uma loja física, por ser uma atividade realizada pela *internet,* algumas informações são mais fáceis de serem obtidas, principalmente no que tange ao comportamento de compra e pagamento dos clientes. Dentre estas informações estão a quantidade de itens comprados, o valor gasto em cada item e quanto tempo o cliente demora para visitar o *e-commerce* ou comprar novamente. Quando combinados, todos esses dados podem fornecer à empresa um conhecimento útil acerca de seus clientes.

Pode-se perceber que o comércio eletrônico se mostra como um nicho lucrativo, repleto de oportunidades. Portanto, é fundamental que os *players* que trabalham com este canal, além de grandes redes de *marketplace,* conheçam e entendam o perfil de seus clientes, para propor ações de retenção e rentabilização destes. Surgem então, modelos, técnicas e ferramentas para a clusterização de clientes.

1.1 JUSTIFICATIVA

Em um cenário onde o *e-commerce* vem ganhando cada vez mais relevância, entender o perfil de seu consumidor para traçar estratégias mais assertivas serve como vantagem competitiva para qualquer empresa frente à concorrência. Este canal de vendas, além de lucrativo, é capaz de fornecer dados úteis que, na era do *Big Data*, podem ser traduzidos em informações valiosas.

Uma das abordagens mais comuns para se entender como os clientes se relacionam com o *e-commerce* é o modelo baseado em recência, frequência e valor, também conhecido como modelo RFV. Por isso, essas três métricas são utilizadas como base na clusterização dos clientes.

Mais do que apenas classificá-los utilizando alguma técnica, é importante agrupar os perfis semelhantes. Neste trabalho, isso ocorre por meio de dois algoritmos de clusterização, o *k-means* e o AGNES, uma alternativa mais científica para identificação dos grupos afins.

Em suma, este trabalho servirá como uma possibilidade para se obter conhecimento de clientes de um *e-commerce*, a partir de técnicas de análise de dados. Assim, é possível direcionar ações de *marketing* mais adequadas para os grupos de clientes encontrados, respeitando as diferenças entre eles.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo geral deste trabalho é propor o uso de dois algoritmos de clusterização, *k-means* e AGNES*,* para agrupar clientes com perfis semelhantes de um *e-commerce*, baseado nas três métricas que compõem o modelo RFV: recência, frequência e valor.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Entender como as variáveis recência, frequência e valor, de clientes de um *e-commerce,* se relacionam entre si na formação de grupos;
- Propor a utilização de dois algoritmos de clusterização de clientes baseados nas variáveis recência, frequência e valor;
- Analisar qual dos algoritmos propostos tem melhor desempenho na clusterização;
- Propor sugestões de ações para cada um dos grupos encontrados pelos métodos.

2 REVISÃO DE LITERATURA

Nesta seção será descrito o modelo RFV, seu funcionamento e suas aplicações encontradas na literatura. Essa segmentação é comumente combinada com outras técnicas de clusterização, como *k-means* e/ou métodos hierárquicos. Por fim, o presente trabalho traz da literatura conceitos sobre o Método do Cotovelo e o Coeficiente de Silhueta, para determinação do número ideal de *clusters* e avaliação de desempenho da clusterização, respectivamente.

2.1 MODELO RFV

O modelo RFV, também conhecido como modelo RFM, é definido por três variáveis principais: **Recência** (do inglês *recency*); **Frequência** (do inglês *frequency*); e **Valor Monetário** (do inglês *monetary value*). Na literatura, essas dimensões podem ser conceituadas de diversas formas, a depender da sua aplicação e do seu objetivo. Para Wei et al. (2010):

- **Recência** é definida pelo número de períodos desde a última compra. Para essa dimensão, é medido um intervalo de tempo entre a compra mais recente e uma data de referência para análise, podendo este intervalo ser medido em dias ou meses. Quanto menor o intervalo, melhor o *score* dessa dimensão. Um cliente com uma recência curta é um cliente que comprou recentemente;
- **Frequência** pode ser definida como o número de compras feitas por um cliente em um intervalo de tempo. Quanto maior a frequência, maior o *score* dessa dimensão. Um cliente com *score* alto representa um cliente que comprou vários itens dessa empresa;
- **Valor** pode ser definido como o montante total gasto pelo cliente em um determinado período de tempo. Também pode ser entendido como o valor médio gasto por compra. Analogamente à dimensão ‘frequência’, quanto maior o valor, maior o *score* dessa dimensão. Um cliente com *score* alto simboliza um cliente que despende um grande valor em compras.

Os *scores* RFV de cada cliente podem ser calculados e tabelados, conforme mostra a Tabela 1. O cliente nº 1 comprou há 3 dias e acumula um total de 6 compras no valor de 540 unidades monetárias. O cliente nº 2, por sua vez, comprou há 6 dias, somando 10 compras no valor total de 940 unidades monetárias.

Tabela 1 - Score RFV por cliente

Código do Cliente	Recência (dias)	Frequência (número)	Valor (u.m.)
1	3	6	540
2	6	10	940
3	45	1	30
4	21	2	64
5	14	4	169
6	32	2	55
7	5	3	130
8	50	1	950
9	33	15	2430
10	10	5	190
11	5	8	840
12	1	9	1410

FONTE: O autor (2020).

Há diversas formas de segmentar clientes, a partir dos seus *scores* RFV. Como mencionado anteriormente, as diversas aplicações desse modelo definem o conceito das dimensões e, portanto, a forma de analisá-las. Ou seja, cabe à empresa definir como irá entender e agrupar comportamentos semelhantes.

Uma das formas de segmentar clientes utilizando o modelo RFV é encontrado em Nimbalkar (2013). Para cada uma das três dimensões, os clientes são ordenados segundo seus respectivos *scores*. Para a ‘recência’ essa ordem é crescente, de modo que os consumidores ficam organizados a partir de quem comprou mais recentemente. Para as demais dimensões, a ordem é decrescente, de modo a manter no topo da lista os clientes que mais compram/gastam. Quando ordenados, os clientes devem ser divididos em quintis (grupos com 20% de clientes).

Wei et al. (2010) atribuem a cada quintil, de cada uma das dimensões, uma pontuação de 1 a 5. Os primeiros 20% recebem nota 5. Os 20% seguintes, nota 4 e assim por diante, até que os últimos 20% recebem nota 1. Ao final desse processo, cada cliente tem um *score*, composto pela pontuação recebida nas três dimensões. Por exemplo, clientes com a pontuação ‘544’ representam um grupo com a recência curta (5) e com frequência (4) e valor (4) altos. Analogamente, um cliente com a pontuação equivalente a ‘132’, tem uma recência (1) alta, frequência (3) média e valor (2) baixo. O conceito de ‘alto’ e ‘baixo’ é subjetivo e também é variável de empresa para empresa, de acordo com seus objetivos na segmentação RFV.

Tabulando as possibilidades, tem-se 125 combinações possíveis e, portanto, 125 estratégias diferentes para se adotar para cada combinação ou, no melhor dos casos, pode-se pensar em aplicar as mesmas estratégias para combinações similares a fim de reduzir a complexidade do problema. É nesse momento que surgem as variações e adaptações. Pode-se, por exemplo, dividir cada dimensão em tercis e adotar apenas 3 notas (sendo 3 a máxima e 1 a mínima). Assim, tem-se 27 combinações.

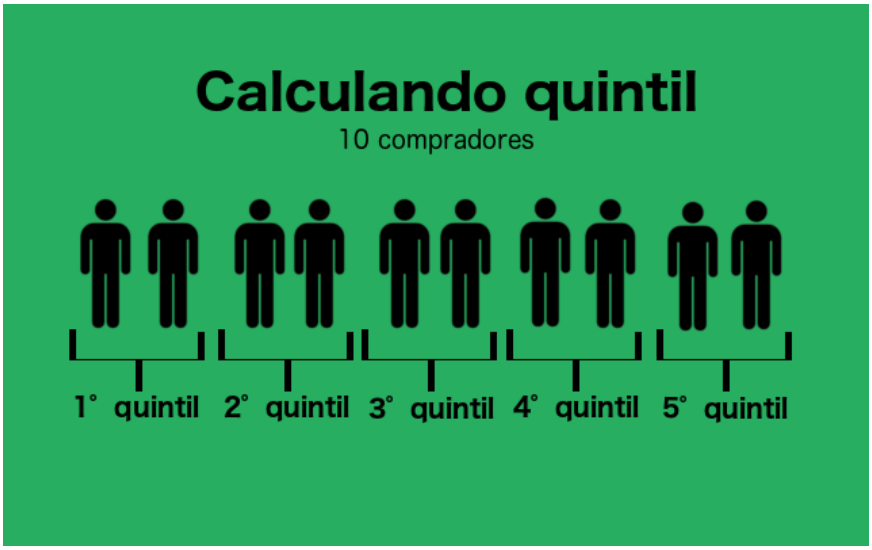
Nimbalkar (2013), em seus estudos, divide a modelagem RFV em 6 passos: na primeira etapa, de *Pré-processamento de Dados*, ocorre a exclusão de dados nulos ou incorretos e a transformação/normalização de dados; na segunda etapa, de *Análise RFV*, as dimensões são ordenadas conforme citado anteriormente e divididas em quintis. Nesta etapa é possível atribuir um peso a cada dimensão, adaptando para a realidade da empresa qual das variáveis tem mais significado; na terceira etapa, de *Clusterização*, o agrupamento é realizado por meio do método *k-means*, que será explicado ao longo deste trabalho*;* na quarta etapa, de *Classificação*, ocorre a classificação e diferenciação dos *clusters*, que pode ser feita de diversas formas; na quinta etapa, de *Gerar Regras de Associação*, acontece uma análise de padrões que mais se repetem e; na última etapa, de *Evolução de Resultados*, é o momento de analisar se os *clusters*, métodos de clusterização e as estratégias definidas para cada grupo foram funcionais ou não.

A descrição do modelo feita por Vasconcelos (2017) é bastante parecida com o explicado anteriormente. O primeiro passo, segundo o autor, é identificar os valores em cada uma das dimensões RFV. Uma diferença da abordagem anterior aparece na segunda etapa, de atribuição da pontuação. Para o autor, há dois métodos de cálculos e regras que podem ser utilizados. O primeiro é a *Pontuação por Quintis*, processo análogo ao mostrado anteriormente. Em resumo:

- Quanto mais recente for a compra do cliente, maior será o *score* de R;
- Quanto mais compras o cliente realizar, maior será a pontuação de F;
- Quanto maior for o gasto dele, maior será o score de V.

Os clientes são organizados, de acordo com seu *score*, em ordem crescente para a ‘recência’ e ordem decrescente para ‘frequência’ e ‘valor’. Então são divididos em quintis para cada variável, como mostra a Figura 1. Para o caso de 10 clientes, cada quintil possui 2 deles, 20% da população. A pontuação segue a escala de 1 a 5, como mencionado anteriormente.

Figura 1 - Exemplo de divisão em quintis



FONTE: VASCONCELOS (2017).

O segundo método é a *Pontuação por Inferência*. Segundo o autor, este método é totalmente baseado em decisões e regras do negócio. Deve-se tomar cuidado pois, embora seja mais fácil de pontuar, essa pontuação pode estar enviesada, principalmente se for atribuída por achismos e não dados.

Por fim, há a etapa de segmentação de clientes. São utilizados os conceitos de Putler (2017). A partir dos *scores* de R e da média dos *scores* F e V, os clientes são classificados conforme a Figura 2.

Figura 2 - Pontuações RFV

Nome do segmento	Intervalo do valor de R	Média de F e V
Champions	4 a 5	4 a 5
Loyal Customers	2 a 5	3 a 5
Potential Loyalist	3 a 5	1 a 3
New Customers	4 a 5	0 a 1
Promising	3 a 4	0 a 1
Customers Needing Attention	2 a 3	2 a 3
About to Sleep	2 a 3	0 a 2
At Risk	0 a 2	2 a 5
Can't Lose Them	0 a 1	4 a 5
Hibernating	1 a 2	1 a 2
Lost	0 a 2	0 a 2

FONTE: VASCONCELOS (2017).

Para cada segmento há a descrição do respectivo tipo de cliente, na Figura 3. Em Putler (2017) há sugestões de ações para cada um dos segmentos.

Figura 3 - Descrição dos Segmentos de Clientes

Segmento	Tipo de cliente
Champion	Comprou recentemente, compra com frequência e gasta muito
Loyal Customer	Gasta bem e frequentemente. É responsivo a promoções
Potential Loyalist	Comprador recente, gasta uma boa quantia e já comprou mais de uma vez
New Customer	Comprou recentemente, mas, não compra com frequência
Promising	Comprou recentemente, mas, em geral não gasta muito
Needing Attention	Valores de R, F e M acima da média, porém, podem não ter comprado recentemente
About to Sleep	Abaixo da média de R, F e M. Podemos perdê-lo se não reativá-lo
At Risk	Gastou muito dinheiro e comprou frequentemente, mas, há muito tempo não compra
Can't Lose Them	Gastou muito dinheiro, mas, não fez compras recentemente ou com frequência
Hibernating	Comprou há muito tempo, comprou poucas vezes e gastou pouco
Lost	Menores *scores* de Recência, Frequência e Monetaridade

FONTE: VASCONCELOS (2017).

2.1.1 Aplicações do modelo RFV

Na literatura, muitos autores apontam o modelo RFV como uma importante ferramenta de CRM (*Customer Relationship Management).* Para Nikumanesh et al. (2014), entender o que o consumidor precisa para agregar valor em seus serviços é um fator que afeta o sucesso ou fracasso de uma companhia. Nesse sentido, os modelos de segmentação baseados em recência, frequência e valor são muito utilizados em estratégias de *marketing* de relacionamento por fornecer informações para construção e manutenção de relacionamentos duradouros e lucrativos (DANTAS et al, 2004).

Segundo Nimbalkar (2013), um CRM efetivo deve ser capaz de traçar estratégias para atração de novos clientes e de retenção de clientes antigos. Nesse contexto de retenção, Wei et al. (2010) destacam a importância de conhecer os variados perfis de clientes, para encontrar os mais rentáveis em termos financeiros e então alocar recursos para estratégias de retenção.

Para Putler (2017), em vez de atingir 100% do público-alvo com campanhas, é preferível identificar e segmentar apenas grupos de clientes específicos que serão mais lucrativos para os negócios.

Pinho (2009) diz que estratégias baseadas em RFV buscam métricas ou regras para avaliar o comportamento e valor do cliente para a empresa. Para o autor, clientes com baixa recência, alta frequência e alto valor apresentarão um alto CLTD (*customer life time duration)*, e estarão dispostos a manter um vínculo contínuo com a empresa. Diz ainda que, contrariamente ao caso anterior, clientes com alta recência, baixa frequência e baixo valor, são mais propensos à interrupção do vínculo empresarial, respondendo pior a campanhas de *marketing*, pois já se encontram no fim do CLTD.

Muitas empresas usam o RFV em seu time de *Customer Success*, para mapear e classificar seus compradores, com o objetivo de fidelizar e recompensar quem já ama a marca e reativar quem está prestes a deixar de usá-la, munindo-se de uma técnica chamada *Targeting Marketing.* Nessa técnica, campanhas e ações são direcionadas a pessoas que fazem parte de um mesmo segmento, visando assim uma taxa de conversão maior e, consequentemente, maior receita (VASCONCELOS, 2017).

Em suma, o modelo RFV mensura quando os clientes compram, quanto compram e quanto gastam em suas compras. Esse histórico de comportamento ajuda a predizer um comportamento futuro (WEI et al., 2010). O texto de Putler (2017) ressalta que a segmentação só funciona quando se tem clientes com histórico. Para os que compram apenas uma vez, o modelo não é o mais adequado. Além disso, são necessários *softwares*/ferramentas para calcular os *scores*/pontuações, dependendo do volume de clientes. Hoje em dia, há plataformas desenvolvidas exclusivamente para acompanhar a evolução RFV dos compradores.

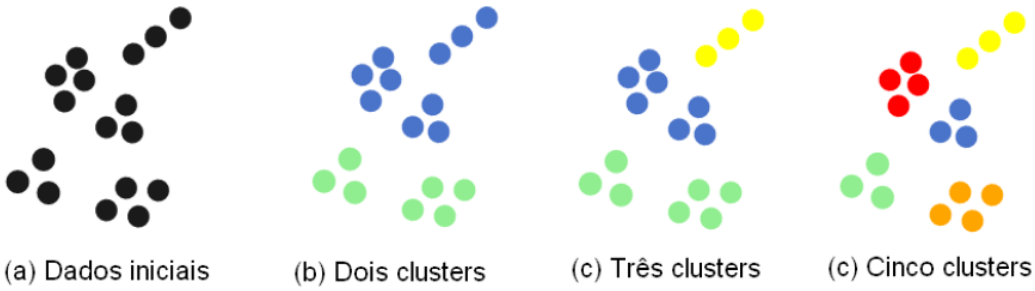
Por fim, vale destacar que a literatura ainda é carente de estudos contendo aplicações da clusterização baseada no RFV, que não em ações de CRM. A maioria dos artigos ou trabalhos publicados voltam suas análises para essa área do *marketing*, embora existam alguns poucos estudos voltados para outras áreas. Nikumanesh (2014), por exemplo, analisa a aplicação num banco do agronegócio no Irã. Wei et al. (2010) citam outros exemplos, como as aplicações em indústrias de telecomunicações e organizações financeiras.

2.2 CLUSTERIZAÇÃO DE DADOS

"*Data is the new oil",* em tradução livre, "Dados são o novo petróleo", é uma expressão muito utilizada atualmente por cientistas de dados, executivos de grandes corporações e economistas. A grande questão nesse tema é como trabalhar com a enorme quantidade de dados existentes. Nesse contexto, a clusterização surge como uma importante técnica para análise e interpretação de dados. Mas afinal, o que é esse método?

Entende-se por clusterização a técnica de dividir uma amostra ou população em grupos, denominados *clusters*. Esse agrupamento é baseado na similaridade entre os elementos de modo que os dados de um mesmo *cluster* tenham mais características em comum entre si do que com dados de outros *clusters* (KLEINA, 2015). A Figura 4 representa em (a) os dados iniciais brutos. Em (b), (c) e (d), representa um exemplo de agrupamento de dados em dois, três e cinco *clusters* respectivamente.

Figura 4 - Clusterização de dados

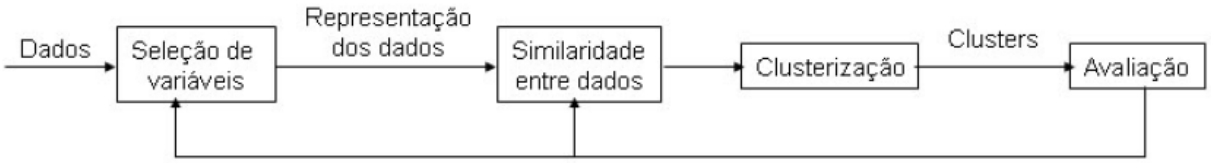


FONTE: OLIVEIRA (2008).

É importante entender que os conceitos de 'classificação' e 'clusterização' são diferentes. A diferença é que na primeira técnica, os dados devem ser atribuídos a grupos já conhecidos previamente, enquanto a segunda deve "descobrir" esses grupos (OLIVEIRA, 2008). Assim, a clusterização é um processo de aprendizado não supervisionado, uma vez que não existem exemplos pré-definidos que evidenciem que algum tipo de relação deva existir entre os dados (KLEINA, 2015). Isso significa que tanto o número ótimo de *clusters* quanto as características que revelam semelhanças ou diferenças são definidas pelo próprio processo (BOSCARIOLI, 2008).

Para Oliveira (2008), há 4 etapas principais para executar o processo de clusterização, mostradas na Figura 5. Vale ressaltar que na literatura não há um consenso acerca dessas etapas e que cada autor define conceitos e escopos diferentes para cada uma delas.

Figura 5 - Etapas da clusterização



FONTE: OLIVEIRA (2008).

A **primeira etapa** é a mais simples, que se resume na definição de variáveis e atributos mais relevantes do conjunto inicial de dados (OLIVEIRA, 2008). Por vezes, um pré-processamento de dados se faz necessário (BOSCARIOLI, 2008). Essa seleção inicial varia de estudo para estudo, conforme seus objetivos.

2.2.1 Medidas de Similaridade

Cassiano (2014) sintetiza a ideia básica da clusterização, que é: elementos que compõem o mesmo *cluster* devem apresentar alta similaridade, mas devem ser muito dissimilares de objetos de outros *clusters*. Para a autora, toda clusterização é feita com objetivo de maximizar a homogeneidade dentro de cada *cluster* e maximizar a heterogeneidade entre *clusters*.

Nesse contexto, está a **segunda etapa** da clusterização descrita por Oliveira (2008), que trata da similaridade entre dados/*clusters*, na qual devem-se encontrar maneiras de quantificar a similaridade ou dissimilaridade entre eles. Assim, faz-se necessário analisar distâncias *intragrupos* e *intergrupos*. Uma baixa distância *intragrupo*, ou seja, entre os dados de um mesmo grupo, indica um bom arranjo de agrupamento, pois esses dados são altamente relacionados entre si. Uma alta distância *intergrupo*, ou seja, entre *clusters* ou dados de *clusters* diferentes, é desejável, porque indica que os grupos individuais são, na maior parte, independentes entre si (BOSCARIOLI, 2008).

Quando se fala da **proximidade entre dados**, alguns métodos partem de uma matriz que reflete de maneira quantitativa essa proximidade. A matriz pode ser definida como *matriz de proximidade*, *matriz de similaridade* ou ainda, *matriz de dissimilaridade*, a depender da relação entre seus dados.

A proximidade ou distância entre dois dados, x_i e x_j, pode ser denotada por $d(x_i, x_j)$ e deve atender a algumas propriedades (BOSCARIOLI, 2008; CASSIANO, 2014; OLIVEIRA, 2008):

- Positividade: $d(x_i, x_j) \geq 0$;
- Simetria: $d(x_i, x_j) = d(x_j, x_i)$;
- Reflexiva: $d(x_i, x_j) = 0 \iff i = j$;
- Desigualdade Triangular: $d(x_i, x_j) \leq d(x_i, x_h) + d(x_h, x_j)$.

Quanto menor o valor de $d(x_i, x_j)$, mais semelhantes serão os objetos e de acordo com um critério, ficarão no mesmo *cluster*. De outro modo, quanto maior a distância, menos similares serão os objetos e, em consequência, eles deverão estar em grupos distintos, pelo mesmo critério (CASSIANO, 2014).

Dentre os cálculos de similaridades, uma das distâncias mais utilizadas é a *distância de Minkowski*, definida pela equação 1.

$$d(x_i, x_j) = \sqrt[p]{\sum_{k=1}^{d} \left(|x_{ik} - x_{jk}|\right)^p}, \qquad p \geq 1 \tag{1}$$

Quando $p = 1$, tem-se a *distância de Manhattan* ou *distância City-Block*, expressa na equação 2.

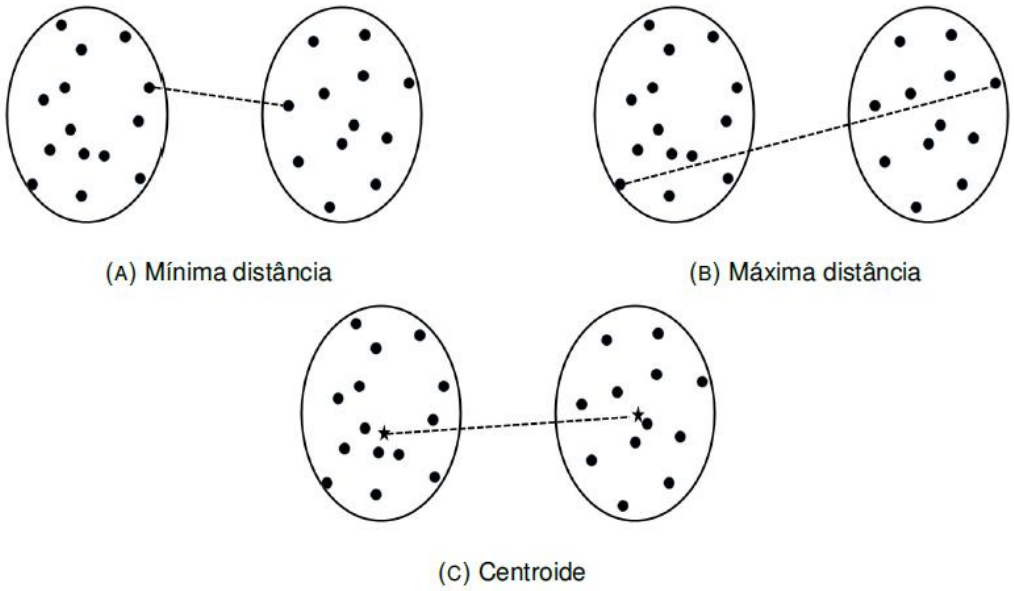
$$d(x_i, x_j) = \sum_{k=1}^{d} \left(|x_{ik} - x_{jk}|\right) \tag{2}$$

Por fim, quando $p = 2$, tem-se a *distância Euclidiana*, que é a mais utilizada e definida pela equação 3.

$$d(x_i, x_j) = \sqrt[2]{\sum_{k=1}^{d} \left(|x_{ik} - x_{jk}|\right)^2} \tag{3}$$

No que tange à **proximidade entre clusters**, uma das maneiras de medi-la é calculando a distância entre todos os pares de pontos dos clusters, em que cada ponto pertence a um *cluster* distinto. Se a distância mínima entre todos os pares de pontos é escolhida, tem-se a mínima distância (*single linkage)*; se for escolhida a distância máxima entre todos os pares de dados, tem-se a máxima distância (*complete linkage)*; e se for escolhida a distância entre os centroides dos *cluster*s, tem-se a distância pelo centroide (*average linkage)* (BOSCARIOLI, 2008; KLEINA, 2015). A Figura 6 ilustra as três distâncias mencionadas.

Figura 6 - Medidas de proximidade entre *clusters*, pelo método (A) mínima distância, (B) máxima distância e (C) centroide



FONTE: Adaptado de KLEINA (2015).

Vale ressaltar que a aplicação de um método ou outro para o cálculo da distância entre *clusters* depende do formato dos grupos (BOSCARIOLI, 2008; KLEINA, 2015). Isso é importante pois, em alguns algoritmos, a medida entre *clusters* é necessária (para unir grupos similares, por exemplo). Se aplicado um método qualquer, os algoritmos podem unir os *clusters* de maneira errada. Segundo Baeza-Yates e Frakes (1992) apud Boscarioli (2008) a máxima distância é melhor para *clusters* mais compactos, e de acordo com Rocha e Lanchi (2005) apud Boscarioli (2008) a mínima distância gera *clusters* mais alongados.

2.2.2 Técnicas de Clusterização de Dados

Para Oliveira (2008), é na **terceira etapa** do processo de clusterização que se define o modo de agrupamento de dados. Na literatura, há vários algoritmos e técnicas de clusterização, que levam em consideração diversos aspectos, tais como: a forma com que os dados estão sendo representados, como medir a similaridade entre dados e entre *clusters* (discutido na seção anterior) e como estimar a qualidade do resultado obtido pelo método. A definição de diferentes algoritmos de clusterização se dá pela maneira como estes aspectos são abordados juntamente com a escolha dos parâmetros iniciais (KLEINA, 2015; OLIVEIRA, 2008).

Cada algoritmo funciona de uma maneira e, portanto, encontra um resultado diferente para a clusterização. Conhecendo-os, é possível determinar qual deles mais se adequa aos objetivos estabelecidos.

Os vários algoritmos existentes, bem como as suas variações, podem ser classificados de acordo com as diferentes técnicas que empregam no agrupamento de dados. Como destacado por Cassiano (2015), alguns algoritmos de clusterização integram as ideias de vários outros, então, algumas vezes, é difícil classificar um dado algoritmo como unicamente pertencendo a somente uma categoria de método de clusterização. Além do que, algumas aplicações podem ter critérios de clusterização que requerem a integração das várias outras técnicas. Por fim, vale ressaltar que não há um consenso na literatura quanto às classificações que serão descritas nesta seção. Embora parecidas, as classificações variam de autor para autor.

Uma classificação inicial divide os algoritmos segundo a abordagem *hard* e a abordagem *fuzzy*. A primeira, determina que cada objeto pertence completamente a um único *cluster.* A segunda, por sua vez, permite que um objeto seja classificado em vários grupos, com diferentes graus de pertinência a cada um deles (KLEINA, 2015).

2.2.2.1 Métodos de Clusterização do Tipo *Hard*

As técnicas do tipo *Hard*, cuja abordagem determina que um objeto pertence ou não a um dado agrupamento, é dividida em várias categorias. Serão destacadas cinco destas, descritas a seguir:

2.2.2.1.1 Algoritmos por Particionamento

Os métodos de clusterização baseados em particionamento dividem uma única vez os dados (n) em um número (k) determinado de *clusters*. Particionar uma única vez pode ser vantajoso quando se tem muitos dados e o armazenamento e processamento de todas as possibilidades de divisões da clusterização hierárquica (descrita adiante) torna-se computacionalmente cara (KLEINA, 2015).

Os algoritmos por particionamento têm por propósito maximizar a similaridade entre elementos de um mesmo *cluster* e minimizar a similaridade entre elementos de *clusters* diferentes, pela otimização de uma função objetivo. Esta função pode representar critérios diferentes a serem otimizados: um critério global usa um ponto representativo de cada *cluster* e agrupa os demais de acordo com a similaridade dos dados com este ponto representativo, já um critério local organiza os grupos pelas informações estruturais dos dados, como por exemplo atribuir um elemento e seus vizinhos mais próximos a um mesmo *cluster* (KLEINA, 2015; OLIVEIRA, 2008).

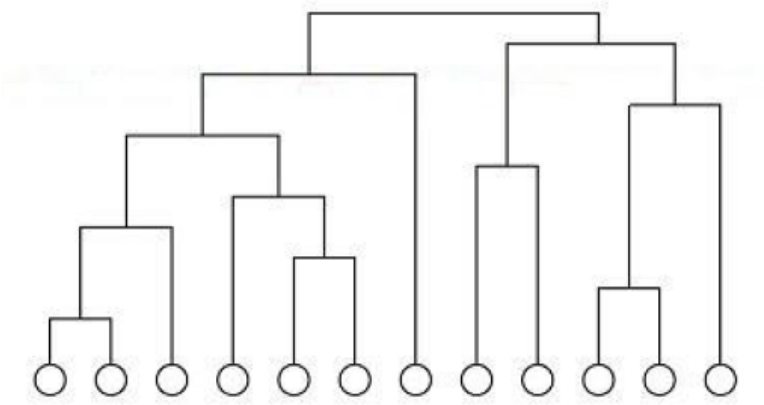
Os métodos particionais produzem agrupamentos simples e são efetivos se o número (k) de *clusters* puder ser razoavelmente estimado, se os *clusters* são de forma convexa e possuem tamanho e densidade similares. O algoritmo utiliza uma estratégia iterativa de controle para determinar que objetos devem mudar de *cluster*, de forma que a função objetivo usada seja otimizada e termina quando não existem atribuições possíveis capazes de melhorar esta função objetivo (CASSIANO, 2014).

Dentre os algoritmos de particionamento mais utilizados estão o *k-means* (algoritmo baseado em centroides) e o *k-medoids (*algoritmo baseado em medoides). O primeiro método será descrito adiante, na Seção 2.3.1.

2.2.2.1.2 Algoritmos Hierárquicos

Nesta classe de algoritmo, os dados são organizados em uma estrutura hierárquica de acordo com a proximidade entre eles (CASSIANO, 2014). Essa decomposição hierárquica é representada por um dendrograma (FIGURA 7), uma espécie de árvore que mostra as decorrentes divisões ou uniões dos dados nos *clusters*. As folhas da árvore simbolizam os dados e conforme a árvore vai crescendo, os dados vão se agrupando para formar grupos maiores, até que todos sejam unidos em um único *cluster*, representado pela raiz (KLEINA, 2015).

Figura 7 - Dendrograma

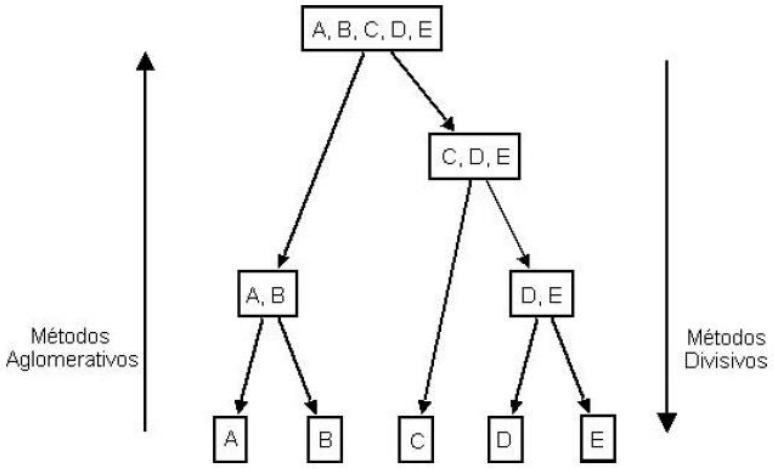


FONTE: Adaptado de OLIVEIRA (2008).

Segundo Carlantonio (2001) apud Kleina (2015), o dendrograma pode ser construído de duas maneiras (FIGURA 8):

- Aglomerativo (*bottom-up*): parte-se das folhas superiores para a raiz, isto é, inicialmente os dados representam *clusters* unitários e a cada iteração, os dois *clusters* mais similares são unidos, até que no final, reste apenas um único *cluster* contendo todos os dados;
- Divisivo (*top-down*): parte-se da raiz para as folhas. Todos os dados começam aglomerados em um único *cluster* e a cada etapa um *cluster* é selecionado e dividido em dois *clusters*. No final, tem-se tantos *clusters* quanto o número de dados originais.

Figura 8 - Métodos Aglomerativos e Divisivos



FONTE: Adaptado de OLIVEIRA (2008).

Os métodos aglomerativos hierárquicos requerem o cálculo da matriz de dissimilaridades entre os grupos, visto que inicialmente cada elemento é um *cluster.* Então, o agrupamento entre elementos, elementos e grupos ou entre grupos se dá por meio da adoção de algum critério em relação à distância. Após cada agrupamento, recalcula-se a matriz de dissimilaridade de acordo com os novos grupos formados e repete-se o processo até que todos os elementos estejam em um único grupo. (CARLANTONIO, 2001 apud KLEINA, 2015). O algoritmo AGNES (*Agglomerative Nesting*) é possivelmente o método aglomerativo hierárquico mais conhecido.

Já os métodos divisivos hierárquicos fazem o caminho contrário dos métodos aglomerativos. O algoritmo DIANA (*Divisive Analysis)* inicia com todos os elementos em um único *cluster*, e em cada passo, o *cluster* com maior diâmetro é dividido em dois *clusters*. O diâmetro de um *cluster* é definido como a dissimilaridade máxima entre todos os elementos dentro de um *cluster*. Recalcula-se a dissimilaridade entre os *clusters* e repete-se o processo até que cada novo *cluster* contenha somente um elemento (CARLANTONIO, 2001 apud KLEINA, 2015).

Embora bastante simples, os métodos hierárquicos não são capazes de efetuar ajustes uma vez que uma união ou divisão tenha sido feita. Esse fato pode comprometer o resultado final visto que, quando um conjunto de dados é unido ou particionado, a próxima iteração será processada com base nos grupos recém-formados, de modo que uma má união de *clusters* não pode ser corrigida nos passos seguintes. Outra desvantagem desse tipo de método é a sensibilidade com respeito a ruídos e também o alto custo computacional quando grandes conjuntos de dados são utilizados (BOSCARIOLI, 2008). Apesar disso, esses métodos são muito utilizados quando há necessidade de gerar hierarquia entre dados como, por exemplo, dados de pesquisa sobre evolução de espécies (CASSIANO, 2014).

Diferentemente dos algoritmos de particionamento, os métodos hierárquicos não necessitam do número de *clusters* como parâmetro de entrada, porém uma condição de término deve ser adotada indicando quando o processo de divisão ou união de grupos deve acabar (CARLANTONIO, 2001 apud KLEINA, 2015).

Dentre os algoritmos supracitados, o AGNES, de abordagem aglomerativa, será objeto de estudo para fins de comparação com o *k-means*. O método será melhor descrito adiante, na Seção 2.3.2.

2.2.2.1.3 Algoritmos baseados em Densidade

Os métodos baseados em densidade diferem-se entre si pela forma com que crescem os *clusters*: uns determinam os *clusters* de acordo com a densidade da vizinhança dos objetos, outros, trabalham de acordo com alguma função de densidade (CASSIANO, 2014). Os *clusters* representam regiões com alta densidade separados por regiões de baixa densidade, que são os ruídos. Os agrupamentos são definidos matematicamente pela identificação de pontos mais densos. Portanto, para cada dado de um *cluster*, a vizinhança deve conter um número mínimo de objetos (BOSCARIOLI, 2008; CARLANTONIO, 2001 apud KLEINA, 2015).

Essa classe de métodos tem a vantagem de formar grupos de formas arbitrárias capazes de filtrar ruídos. Outro ponto positivo é que o método não necessita do número de *clusters* como parâmetro inicial (CASSIANO, 2014; OLIVEIRA, 2008). Existem diversos métodos capazes de identificar áreas de densidade. Dentre os mais conhecidos, estão o método DBSCAN (*Density Based Spatial Clustering of Applications with Noise*) e o método OPTICS (*Ordering Points To Identify the Clustering Structure*), que é uma extensão do primeiro.

2.2.2.1.4 Algoritmos baseados em Grade

Os métodos dessa classe utilizam uma estrutura de dados em grade (*grid)* de multiresolução. Eles particionam o espaço de objetos em um número finito de células que formam uma estrutura de grade, na qual todas as operações de clusterização são efetuadas. Considera-se que todos os dados que estejam em um mesmo grupo também estejam na mesma célula de grade. Dessa forma, os dados pertencentes a uma mesma célula podem ser representados e tratados como um único dado (BOSCARIOLI, 2008; CASSIANO, 2014; KLEINA, 2015). A principal vantagem desta abordagem é seu tempo de processamento rápido.

Há dois métodos mais conhecidos dentre os algoritmos baseados em grade: o método STING (*Statistical Information Grid)* e o método CLIQUE (*Clustering In Quest*), que é mais adequado para agrupar dados de alta dimensão em ampla base de dados, combinando métodos de clusterização baseados em grade e em densidade (CARLANTONIO, 2001 apud KLEINA, 2015).

2.2.2.1.5 Algoritmos baseados em Modelo

São métodos que utilizam um modelo de referência para cada *cluster*. Os algoritmos de agrupamentos baseados em modelos têm por meta otimizar o ajuste entre os dados de entrada e algum modelo matemático, isto é, um modelo pressuposto é elaborado para cada *cluster* e encontra-se o melhor ajuste do dado ao modelo. Uma função densidade que dita a distribuição espacial dos dados pode ser usada para localizar grupos. Estes métodos geralmente são baseados na teoria de que os dados são gerados por uma mistura de distribuições de probabilidades (BOSCARIOLI, 2008; CASSIANO, 2014).

Dentro desta classe de clusterização, há duas subclasses: conceitual e rede neural. A subclasse **conceitual** é uma forma de agrupamento em aprendizagem de máquina que, dado um grupo de objetos não rotulados, produz um projeto de classificação sobre eles. É um processo em duas fases: primeiro é realizado o particionamento e depois é feita uma caracterização dos dados, gerando conceitos e descrições dos grupos. Assim, diferentemente das demais abordagens de agrupamento, não somente identifica grupos como também gera descrições de cada um deles (BOSCARIOLI, 2008; CASSIANO, 2014). Como exemplo nesta categoria, pode-se citar o método COBWEB.

Na subclasse **rede neural** cada grupo é representado por meio de um modelo ou exemplar, que não necessariamente corresponde a um dos objetos do grupo. Por meio de alguma medida de distância, novos objetos são atribuídos ao grupo cujo modelo lhe seja mais próximo (BOSCARIOLI, 2008; KLEINA, 2015). As redes neurais são apropriadas para tarefas de percepção como o reconhecimento, classificação e auto associação de padrões. Apresentam algumas vantagens, tais como: robustez ao ruído, capacidade de generalização, aprendizado adaptativo a partir de exemplos e processamento paralelo (CASSIANO, 2014). A auto-organização de mapas é uma abordagem de rede neural para clusterização, sendo o SOM *(Self-Organizing Map)* o método mais clássico.

Além dos algoritmos citados anteriormente, existem vários outros presentes na literatura, resumidos na Figura 9.

Figura 9 - Algoritmos de Clusterização divididos por categoria

Classe		Exemplos de Algoritmos
Particionamento		*k-means*
		k-medoids
		CLARA
		CLARANS
		CLUSTER
		PAM
Hierárquico		AGNES
		DIANA
		CURE
		CHAMELEON
		BIRCH
		ROCK
		CLINK
Densidade		DBSCAN
		OPTICS
		DENCLUE
		GDBSCAN
		DBCLASD
Grade		STING
		CLIQUE
		WAVECLUSTER
		BANG-CLUSTERING
		GRIDCLUST
Modelo	Conceitual	COWEB
		CLASSIT
		AUTOCLASS
		SNOB
		MCLUST
	Rede Neural	SOM
		ART

FONTE: Adaptado de BOSCARIOLI (2008).

2.2.2.2 Métodos de Clusterização do Tipo *Fuzzy*

Os métodos de clusterização do tipo *fuzzy* são métodos 'não *hard'*, ou seja, que permitem associar um indivíduo a todos os *clusters* usando uma função de pertinência. A restrição adotada nesta metodologia é que a soma dos graus de pertinência de um indivíduo aos *clusters* seja igual a 1. Em um algoritmo *fuzzy*, cada *cluster* é um conjunto *fuzzy* de todos os indivíduos (CASSIANO, 2014).

Segundo Kleina (2015), a clusterização do tipo *hard* pode ser inadequada quando há pontos que estão igualmente distantes de dois ou mais grupos, exigindo a designação completa para algum destes grupos, embora exista chance igualitária do ponto pertencer aos demais grupos. A clusterização do tipo *fuzzy* é mais flexível no sentido de que um dado pode pertencer a mais do que um agrupamento ao mesmo tempo, com graus de pertinência ou associação. Graus de pertinência podem expressar o quanto de certeza ou incerteza o dado foi designado ao *cluster* correto.

Cada conjunto *fuzzy* é caracterizado pela sua função de pertinência que é uma curva que define o grau de posse (valor entre 0 e 1) de cada ponto, onde geralmente as funções de pertinência mais utilizadas são a triangular, trapezoidal e gaussiana (BIONDI NETO et al., 2006 apud KLEINA 2015). Uma desvantagem dos algoritmos *fuzzy* em relação a alguns algoritmos *hard* é que é necessário o conhecimento do número de grupos.

O mais popular algoritmo de clusterização dessa classe é o *fuzzy c-means*, melhor que o *k-means (hard)* para evitar estagnação em mínimos locais. É um método de clusterização não hierárquico que proporciona uma partição *fuzzy* de um conjunto de dados em c *clusters*, por meio da minimização de uma função objetivo que mede a adequação entre os dados e os *clusters* (KLEINA, 2015).

2.3 ALGORITMOS DE CLUSTERIZAÇÃO UTILIZADOS NA PESQUISA

Dentre os algoritmos supracitados, nesta seção serão vistos dois deles: o método *k-means*, método por particionamento, e o *AGNES*, método hierárquico.

2.3.1 Método *k-means*

O *k-means* é um algoritmo do tipo não supervisionado, ou seja, apenas os dados são necessários para sua execução, sem a necessidade de se ter um conhecimento prévio da classe à qual cada observação pertence. Embora não garanta precisão, sua simplicidade e velocidade são aspectos muito considerados na hora de escolher este método para algum fim (ARTHUR et al., 2007). Este método também é encontrado na literatura como Algoritmo de *Lloyd*.

Segundo Santana (2017), o objetivo desse algoritmo é encontrar similaridades entre n dados e agrupá-los conforme o número de *clusters* passado pelo parâmetro k, de modo a manter no mesmo grupo os dados mais similares. Analogamente, para Arthur et al. (2007), o objetivo é escolher um número de *clusters* k, a partir de um conjunto de dados, que minimize a distância quadrática total entre cada ponto e o centroide mais próximo. Entende-se por ‘centroide’ o centro de cada *cluster,* que será a média dos valores neste *cluster* (HUGO, 2017). Para Sampaio (2017), o objetivo deste método é minimizar o momento de inércia total, que pode ser calculado como a soma do quadrado das distâncias de cada ponto ao centro do agrupamento.

Santana (2007) descreve o *k-means* em três passos, semelhante ao encontrado em Arthur et al. (2007), Hugo (2017), Kodinariya (2013) e em demais estudos na literatura. A primeira etapa pode ser definida como **Inicialização**. Nesta fase, a partir de um conjunto de dados n, o algoritmo gera de forma aleatória k centroides, onde o número de centroides é representando pelo parâmetro k, arbitrariamente escolhido por quem executa o algoritmo.

A segunda etapa é a **Atribuição ao *Cluster***. A partir dos centroides gerados, são calculadas as distâncias entre todos os pontos de dados. Cada registro será atribuído ao centroide ou *cluster* que tem a menor distância, ou seja, o mais próximo. Esta distância, como visto anteriormente, é calculada por meio de equações e, neste caso, é muito utilizada a distância Euclidiana. Ao final desta etapa, os dados estão divididos conforme o número de centroides estipulados pelo argumento k.

Na terceira etapa ocorre a **Movimentação dos Centroides**. Uma vez que os pontos de dados foram atribuídos aos *clusters* conforme sua distância, o próximo passo é recalcular o valor dos centroides. Nesta etapa é calculada a média dos valores dos pontos de dados de cada *cluster* e o valor médio será o novo centroide. O termo ‘movimentação’, se refere a alteração da localização do centroide (representado por X) em um plano, quando desenhado em um gráfico (FIGURA 10).

Figura 10 - Movimentação dos Centroides

FONTE: Adaptado de SANTANA (2007)

A segunda e terceira etapa são repetidas até o *cluster* se tornar estático ou até que algum critério de parada tenha sido atingido. O *cluster* torna-se estático quando nenhum dos pontos de dados troca de *cluster*. Um critério de parada pode ser o número de iterações máximas que o algoritmo irá fazer durante a fase de otimização.

Em suma, a partir dos centroides iniciais, os dados são repetidamente atribuídos ao *cluster* mais próximo e o valor do centroide é atualizado, sempre tomando o valor médio de todos os pontos naquele *cluster*. Este algoritmo pode ser executado via bibliotecas e pacotes nos principais *softwares* de programação, como o R e o *Python*, e até mesmo no *Excel*.

Sampaio (2017) ressalta que o *k-means* é um algoritmo sensível ao número k de *clusters*. Como a clusterização depende diretamente desse parâmetro, o resultado final está atrelado à escolha de k e dos centroides inicialmente escolhidos. Nesse sentido, há o método *k-means*++, explicado com mais detalhes em Arthur et al. (2007), que basicamente seleciona os dados que serão os centroides iniciais de maneira inteligente. Ademais, há o conhecido Método do Cotovelo, descrito adiante, que otimiza o parâmetro k. O autor destaca a importância de se normalizar os dados antes da clusterização, a depender do conjunto, para que nenhum atributo se sobressaia durante o algoritmo e também chama atenção ao formato dos grupos, que pode interferir na clusterização. Vale lembrar que o *k-means* gera os *clusters* sem identificação, ou seja, caracterizar os grupos cabe a quem executou o algoritmo.

2.3.2 Método AGNES

O AGNES, *Agglomerative Nesting*, é um algoritmo do tipo hierárquico aglomerativo. Conforme citado anteriormente, esse tipo de método é muito utilizado quando há necessidade de gerar hierarquia entre dados como, por exemplo, dados de pesquisa sobre evolução de espécies (CASSIANO, 2014). Diferentemente do método anterior, não necessita que o parâmetro k, o número de *clusters*, seja pré-definido. Há apenas a necessidade que o usuário forneça a matriz de similaridade ou dissimilaridade. Ademais, não é preciso assumir nenhuma premissa em relação à distribuição da base de dados. (TORRES, 2013).

Para Pereira (1993), o método opera basicamente sobre a matriz de similaridades ou de dissimilaridades entre os objetos. O AGNES promove a união de *clusters* dois a dois ao longo do processo, até que reste apenas um. Em suma, a primeira iteração do algoritmo une dois objetos da base de dados que estejam mais próximos entre si formando um primeiro *cluster*. Nas demais iterações, promove-se a união entre dois grupos já formados que estejam mais próximos entre si (TORRES, 2013).

Torres (2013), indo ao encontro de Pereira (1993), descreve o método em quatro passos. No primeiro deles, cada objeto representa um *cluster* unitário. Isso significa que, no início do método, há n grupos formados por um único objeto, sendo n o número total de objetos. Num segundo passo, calcula-se a matriz de similaridade (ou dissimilaridade) e dela se extrai o par de grupos com maior similaridade. Esses dois grupos são unidos num terceiro passo, no qual calculam-se novas medidas de similaridade entre o novo *cluster* formado e os demais, de modo que, ao fim desta etapa, o número de grupos é $n - 1$. Por fim, o quarto passo consiste na iteração do segundo e terceiro passos. As etapas são iteradas $n - 1$ vezes, de modo que todos os objetos terminem em um único grupo.

Conforme já mencionado, os algoritmos hierárquicos utilizam um dendrograma para mostrar as divisões ou uniões dos grupos. Para a construção do dendrograma do AGNES, já que é um método do tipo aglomerativo, parte-se das folhas superiores para a raiz, ou seja, de baixo para cima (*bottom-up*).

O AGNES apresenta algumas variações, gerando 'novos métodos'. Os distintos métodos aparecem na forma pela qual definem o par de grupos com menor dissimilaridade (passo 2) e na maneira pela qual determinam as dissimilaridades entre o novo grupo formado e os grupos restantes (passo 3) (PEREIRA, 1993). A diferença, portanto, dá-se nos cálculos de dissimilaridade intragrupos e intergrupos.

Se a mínima distância é adotada ou o valor mínimo da dissimilaridade entre todos os pares de pontos, a técnica recebe o nome *Simple Linkage* (SLINK) ou Método da Ligação Simples (OLIVEIRA, 2008). Segundo Torres (2013), esta regra apresenta melhores resultados quando busca-se encontrar *clusters* com formato mais alongado. Se a máxima distância é utilizada, ou seja, o maior valor da dissimilaridade entre dois pontos quaisquer dos grupos analisados, a técnica é chamada de *Complete Linkage* (CLINK) ou Método da Ligação Completa (OLIVEIRA, 2008). Esta regra tende a formar *clusters* mais compactos, isto é, resulta num número grande de grupos com diâmetro pequeno (TORRES, 2013). Esta segunda técnica é oposta à primeira.

Ainda, se a distância média é usada (ponto central do grupo), recebe o nome de *Average Linkage* ou Método do Centroide. No método *Ward*, a dissimilaridade entre dois *clusters* baseia-se na distância Euclidiana entre seus centroides, multiplicada por um certo fator (TORRES, 2013). Para Seidel et al. (2008), esse método tende a resultar em agrupamentos de tamanhos aproximadamente iguais devido a sua minimização da variância.

Analogamente ao *k-means,* o AGNES também pode ser executado via bibliotecas e pacotes nos principais *softwares* de programação, como o R e o *Python*. Neste último, por exemplo, pode-se definir como parâmetro *'linkage'* um dos quatros métodos supracitados (*single linkage, complete linkage, average linkage* e *ward).*

2.4 MÉTODOS AUXILIARES UTILIZADOS NA PESQUISA

Após a escolha dos algoritmos *k-means* e AGNES como objeto de estudo deste trabalho, faz-se necessário também ter conhecimento de dois métodos que auxiliam na execução e validação da clusterização gerada por tais algoritmos.

O primeiro deles, chamado Método do Cotovelo, é utilizado para definir o número ótimo de *clusters* k, número este que deve ser passado como parâmetro no início da execução do *k-means*. O segundo método, de nome Coeficiente de Silhueta, pode ser aplicado em ambos os algoritmos e tem por objetivo medir o quão bem uma observação está agrupada, pois estima a distância média entre os agrupamentos.

2.4.1 Método do Cotovelo

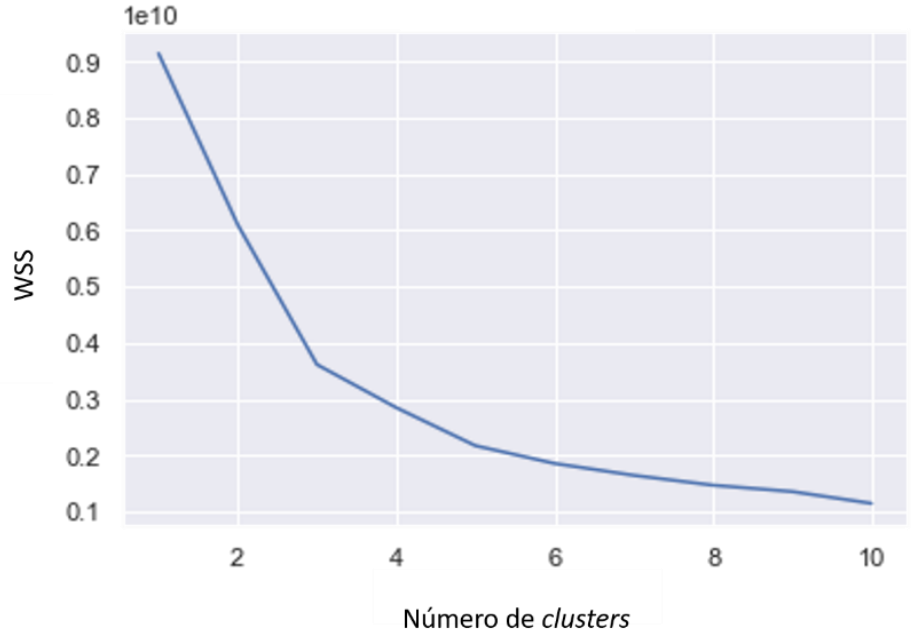
Kodinariya (2013) cita diversos métodos que podem ser utilizados para definir o número de *clusters* no *k-means*, número este passado pelo parâmetro k. Um deles é Método do Cotovelo (traduzido do inglês, *Elbow Method),* considerado pelo autor o método mais antigo. Como o *k-means* calcula a distância das observações até o centroide do agrupamento que elas pertencem, o ideal é que essa distância seja a menor viável. Matematicamente falando, busca-se uma quantidade de agrupamentos em que a soma dos quadrados *intraclusters* ou *wcss* (*within-clusters-sum-of-squares*) seja a menor possível (TEMPORAL, 2017). Geralmente utiliza-se a distância Euclidiana para o cálculo da *wcss*.

O método do Cotovelo é um método visual e seu *output* é um gráfico. O eixo das abcissas, eixo x, representa o número de *clusters* k. O eixo das ordenadas, eixo y, representa o valor da *wcss*. Partindo de $k = 2$, plota-se o seu respectivo valor *wcss*. Os demais pontos do gráfico também são plotados, sempre aumentando o valor de k em uma unidade. Isso significa, que o próximo ponto tem abscissa x (ou k) $= 3$, depois $x = 4$ e assim por diante.

Para um determinado valor de k, o valor da *wcss* cai drasticamente. A partir deste ponto, a variação começa a ser menor, de modo que a *wcss* continua a reduzir, porém, mais vagarosamente. Este é o k procurado (KODINARIYA, 2013).

A curva do gráfico gerado pelo método forma um ‘cotovelo’, que dá origem ao nome do método e indica o número de *clusters* ideal (FIGURA 11). No exemplo, o número de *clusters* ideal, pelo método, está entre 3 e 5.

Figura 11 - Método do Cotovelo



FONTE: o autor (2020).

2.4.2 Coeficiente de Silhueta

Conforme citado anteriormente, Oliveira (2008) descreve o processo de clusterização em quatro etapas, das quais três foram explicadas ao longo deste trabalho. A **quarta** etapa é a Validação de *Clusters*. Os agrupamentos poder ser validados mediante *Índices Internos*, que se baseiam na estrutura intrínseca e nas características do conjunto de dados; *Índices Externos*, que comparam a estrutura dos grupos com uma outra estrutura de grupos previamente conhecida; e *Índices Relativos,* que são usados para comparar diferentes agrupamentos a partir do mesmo algoritmo, mas sob parâmetros diferentes. Um dos índices internos mais populares desde a sua criação é o Coeficiente de Silhueta (WANG et al., 2017).

Partindo do pressuposto que a clusterização tem por objetivo realizar agrupamentos, de modo que dados num mesmo *cluster* sejam muito similares entre si, porém, muito dissimilares quando comparados com dados de outros grupos, o Coeficiente de Silhueta é utilizado para validar tal objetivo. Em suma, este método avalia o quão bem uma observação está agrupada.

Segundo Wang et al. (2017), o Coeficiente de Silhueta analisa a distância de cada ponto de dados para seu próprio *cluster* e o *cluster* vizinho mais próximo. Segundo o *site* da biblioteca *Python Scikit-Learn*, o gráfico de silhueta que é gerado exibe uma medida de quão perto cada ponto de um *cluster* está dos pontos nos *clusters* vizinhos e, portanto, fornece uma maneira de avaliar visualmente parâmetros, tais como o número de *clusters*.

Kassambara (2018) explica que, para cada observação i, o valor da silhueta s_i é calculado a partir de três passos. No primeiro deles, para cada observação i, deve-se calcular a dissimilaridade média a_i entre i e todos os demais pontos do *cluster* ao qual i pertence. No segundo passo, para todos os demais *clusters* C, com exceção do qual i pertence, deve-se calcular a dissimilaridade média $d(i, C)$ de i para todas as observações de C. O menor valor $d(i, C)$ é definido como $b_i = min_C\, d(i, C)$, sendo este valor b_i a dissimilaridade entre i e o *cluster* vizinho mais próximo. Finalmente, o valor do Coeficiente de Silhueta da observação i é definido pela equação 4.

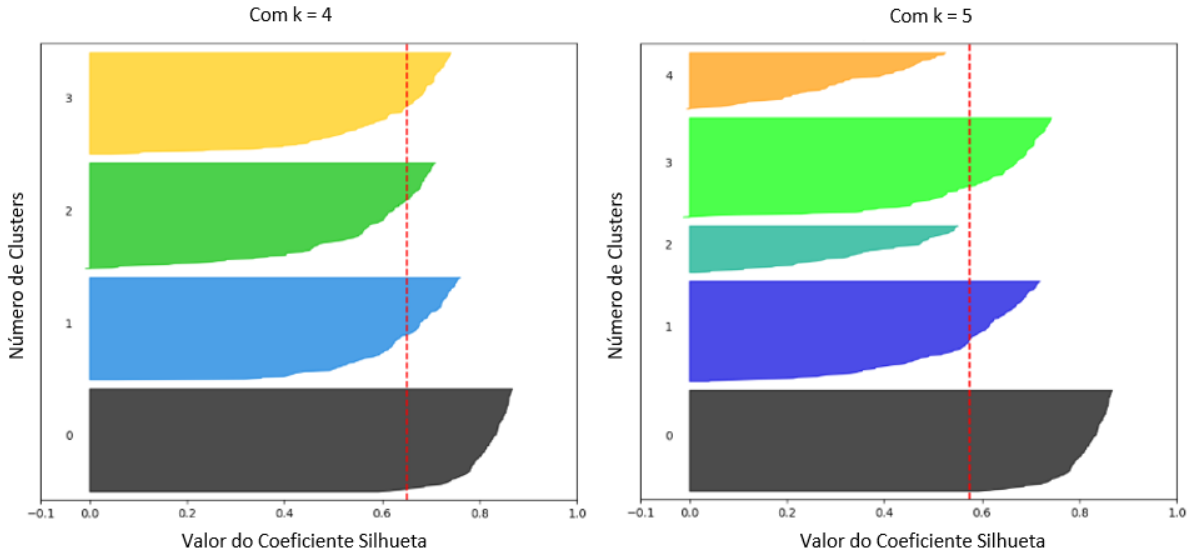
$$S_i = \frac{(b_i - a_i)}{\max(a_i, b_i)} \tag{4}$$

A medida S_i pertence ao intervalo $[-1, 1]$. Coeficientes de silhueta próximos a 1 indicam que a amostra está longe dos aglomerados vizinhos. Isso significa que o resultado do agrupamento é bom. Um valor 0 ou próximo disso indica que a amostra está dentro ou muito perto do limite de decisão entre dois *clusters* vizinhos. Por fim, valores negativos indicam que a amostra pode ter sido atribuída ao *cluster* errado.

A Figura 12 traz os valores dos coeficientes de silhueta de alguns dados. Estes dados foram agrupados em 4 e em 5 *clusters*. É possível perceber que, quando $k = 5$, gráfico da direita, as pontuações de dois grupos (grupo 4 e grupo 2) ficam abaixo da média da silhueta, representado pela linha vermelha.

Ao analisar o gráfico à esquerda, quando $k = 4$, pode-se notar que os coeficientes de todos os grupos estão com valores acima da média da silhueta. Portanto, pode-se concluir que os dados estão melhor agrupados quando $k = 4$.

Figura 12 - Coeficiente de Silhueta

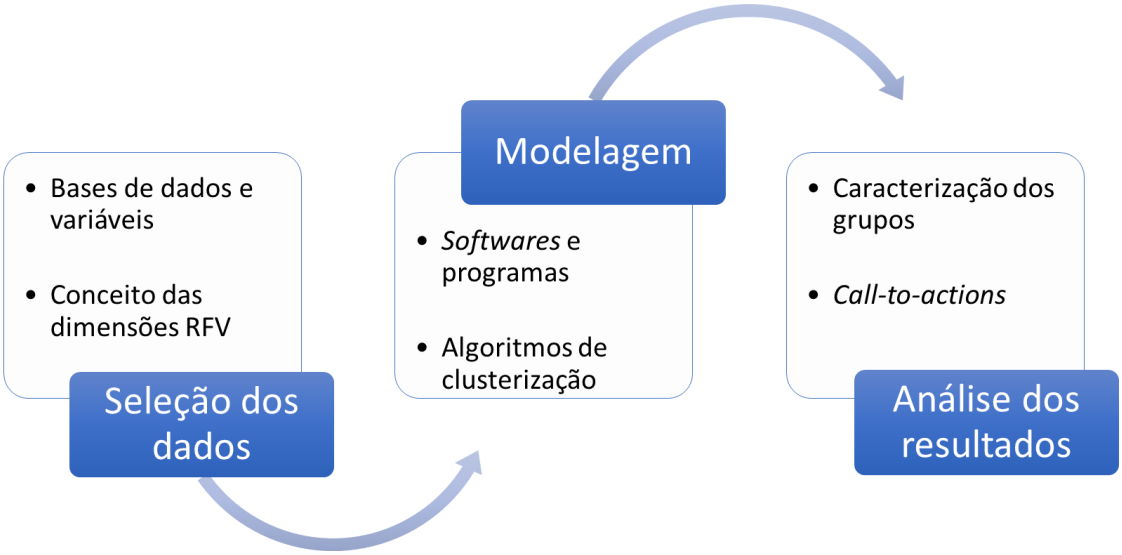


FONTE: o autor (2020).

3 METODOLOGIA

Nesta seção serão descritas as etapas para o desenvolvimento do modelo RFV. Primeiramente, tem-se a seleção das bases de dados e variáveis utilizadas, e o conceito das dimensões RFV que foi adotado neste trabalho. Em seguida, estão os *softwares*, programas e algoritmos de clusterização que compõem a modelagem. Por fim, a análise e interpretação dos resultados, que se dá pela caracterização dos grupos e definição de ações para cada um deles (FIGURA 13).

Figura 13 - Fluxograma metodológico da pesquisa



FONTE: O autor (2020).

3.1 SELEÇÃO DOS DADOS

Os dados utilizados neste trabalho são da *Olist,* um *marketplace* de *e-commerce*, fundado em 2015, em Curitiba, Paraná. As bases de dados foram disponibilizadas pela *Olist* no *Kaggle*, plataforma de hospedagem para projetos e competições de *Data Science*, na qual estão disponíveis também conjunto de dados de outras empresas (OLIST) para diversos estudos.

As bases de dados iniciais, apresentadas na Tabela 2, compreendem informações de cem mil pedidos feitos entre setembro/2016 e setembro/2018, os dados mais recentes disponibilizados pela empresa.

Tabela 2 - Bases de dados da *Olist* no *Kaggle*

Base de dados	**Descrição geral**
olist_customers_dataset.csv	Informações de identificação dos clientes
olist_geolocation_dataset.csv	Informações de geolocalização dos clientes
olist_order_items_dataset.csv	Informações sobre os itens comprados
olist_order_payments_dataset.csv	Informações sobre pagamento
olist_order_reviews_dataset.csv	Informações sobre a avaliação dos usuários
olist_orders_dataset.csv	Informações sobre os pedidos
olist_products_dataset.csv	Informações sobre os produtos
olist_sellers_dataset.csv	Informações sobre as lojas que compõem o *marketplace*
product_category_name_traslation.csv	Tradução para a língua inglesa dos produtos

FONTE: O autor (2020).

Para o desenvolvimento do modelo RFV são utilizadas três das bases supracitadas, a saber:

- *olist_customers_dataset.csv;*
- *olist_orders_dataset.csv;*
- *olist_order_payments_dataset.csv.*

A base *olist_customers_dataset.csv* fornece informações sobre o cliente e sua localização. As variáveis são:

- *customer_id*: identifica os pedidos (cada pedido tem um *id* exclusivo);
- *customer_unique_id*: identificador de um cliente;
- *customer_zip_code_prefix*: primeiros cinco dígitos do CEP do cliente;
- *customer_city:* cidade de origem da compra;
- *customer_state:* unidade federativa (UF) de origem da compra.

Na *Olist*, cada pedido tem um *customer_id*. Isso significa que um mesmo cliente terá diferentes *ids* para diferentes pedidos. Esta variável é a chave de ligação para as demais bases de dados.

Na base *olist_orders_dataset.csv*, por sua vez, estão dispostas informações sobre os pedidos feitos pelos clientes. É composta por:

- *order_id*: identificador exclusivo do pedido;
- *customer_id*: chave para o conjunto de dados anterior;
- *order_status*: referência ao *status* do pedido (entregue, enviado, etc);
- *order_purchase_timestamp*: registro de data e hora da compra;
- *order_approved_at:* registro da data e hora da aprovação do pagamento;
- *order_delivered_carrier_date:* registro da data e hora da publicação do pedido, quando foi entregue ao parceiro logístico;
- *order_delivered_customer_date:* registro da data e hora de entrega do pedido ao cliente;
- *order_estimated_delivery_date:* registro da data de entrega estimada que foi informada ao cliente no ato da compra.

A variável *order_id* é utilizada como chave para o próximo conjunto de dados, olist_order_payments_dataset.csv

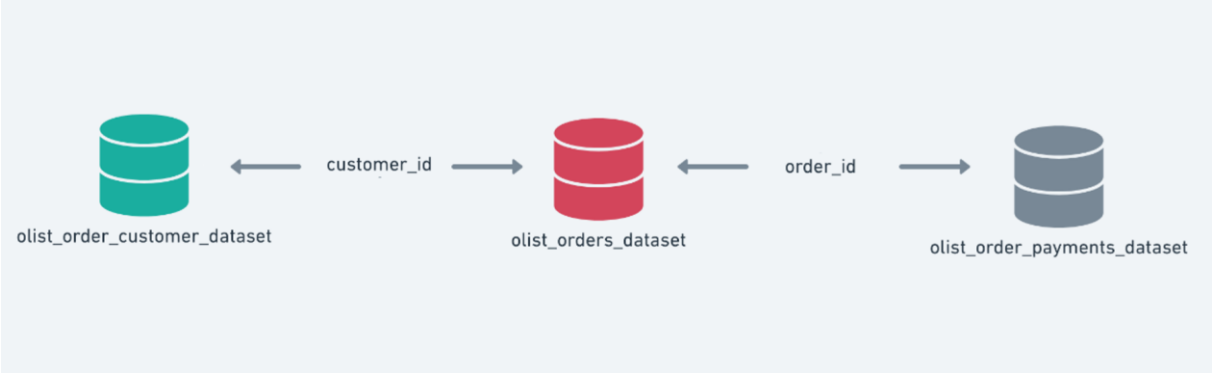
Por fim, a base *olist_order_payments_dataset.csv* traz informações sobre o pagamento das compras feitas. Os dados são:

- *order_id*: identificador exclusivo do pedido (chave com o conjunto de dados anterior);
- *payment_sequential*: sequência criada com todos os meios de pagamento utilizados na compra, caso seja mais de um;
- *payment_type*: método de pagamento escolhido pelo cliente;
- *payment_installments:* número de parcelas escolhidas pelo cliente
- *payment_value*: valor da transação;

A partir dessas três bases, é possível construir uma base única, na qual cada linha representa um cliente, identificado pelo seu *customer_unique_id*, e as colunas representam suas respectivas informações de compras de pedidos e pagamentos. Nesta base não deve haver duplicidade de clientes por linha.

Para o cruzamento correto das três bases, o fluxo deve ser semelhante ao expresso na Figura 14. Partindo da base *olist_customers_dataset.csv*, as informações dos pedidos são agregadas a cada cliente utilizando a base *olist_orders_dataset.csv*. Por meio do identificador do pedido, são agregadas também as informações de pagamento, extraídas da base *olist_order_payments_dataset.csv*.

Figura 14 - Cruzamento de Bases



FONTE: O autor (2020).

Por padrão, convém realizar um tratamento dos dados, excluindo valores nulos ou incorretos. Dessa forma, garante-se que todos os clientes possuem dados corretos para os cálculos das dimensões RFV.

Conforme citado anteriormente, as dimensões RFV podem ser conceituadas de diversas formas, a depender da aplicação e objetivo. Além dos dados das bases da *Olist*, é necessário também fixar uma data de referência. Como o conjunto de dados adotado contém pedidos feitos entre setembro/2016 e setembro/2018, a data de referência pode ser qualquer uma a partir de outubro/2018.

As dimensões, então, podem ser definidas como:

- **Recência:** intervalo em dias entre a data de referência e data da compra mais recente feita pelo cliente;
- **Frequência:** quantidade total de itens comprados a partir da primeira compra até a data de referência;
- **Valor:** valor total gasto por cada cliente a partir da primeira compra até a data de referência.

Com base nestas definições, calcula-se o valor de cada dimensão, para cada um dos clientes. Estas informações são armazenadas em uma base final, que servirá como *input* na modelagem (vide Tabela 1, Seção 2.1). Neste trabalho, os cálculos das variáveis RFV serão feitos diretamente na plataforma de programação escolhida. Porém, poderiam ser feitos numa base auxiliar no *Excel*, ou extraídos de banco de dados.

3.2 MODELAGEM

Diferentemente das abordagens comuns, este trabalho não utiliza a metodologia de dividir os clientes em quintis, a partir de uma ordem, e atribuir a cada dimensão RFV, de cada quintil, um *score* de 1 a 5. Para o agrupamento dos clientes, foram escolhidos dois algoritmos de clusterização, o *k-means* e o AGNES.

Para a construção do modelo e análise dos seus resultados, foi escolhida a linguagem *Python*. A linguagem possui diversos pacotes/bibliotecas que facilitam o desenvolvimento de programas e modelos. As principais utilizadas no trabalho são: *pandas*, para análise e manipulação de dados; *numpy*, para técnicas de álgebra linear; *scikit-learn*, para aprendizado de máquina; e *matplotlib*, para criação e visualização de gráficos. Todo desenvolvimento do modelo será feito em um *jupyter notebook*, uma ferramenta que simula uma IDE (ambiente de desenvolvimento integrado) para diversas linguagens de programação, inclusive *Python*.

Da biblioteca do *scikit-learn,* se extraem os dois algoritmos de clusterização escolhidos. No *k-means,* o parâmetro obrigatório é o k, que representa o número de *clusters*. Este será definido com o auxílio do Método do Cotovelo. Há diversos outros parâmetros que poderiam ser passados ao método *k-means*, mas não são obrigatórios e, portanto, não serão citados. Assim, estes parâmetros recebem seu valor padrão (*default)* do próprio método.

No método AGNES, o parâmetro do número de *clusters* também será o encontrado pelo Método do Cotovelo, para fins de comparação de resultados com o *k-means*. Outro parâmetro importante do algoritmo é o parâmetro *linkage*, que calcula a distância entre os dados dos *clusters*. Por padrão, o algoritmo inicializa com o método '*ward',* e este será mantido.

Além do tratamento de dados feito antes da modelagem, os dados sofrerão um processo de normalização. Isso é importante para o processo, pois as variáveis estão em diferentes escalas e unidades e, dessa forma, uma das dimensões poderia se sobressair diante das outras no momento da clusterização. A normalização garante que isso não acontece. Este processo é executado via biblioteca *scikit-learn*.

Após a execução dos algoritmos, os resultados de ambos podem ser comparados pelo Coeficiente de Silhueta, que também é calculado por meio da biblioteca *scikit-learn*. Com esse coeficiente, é possível saber qual dos métodos foi mais eficiente na clusterização dos dados da *Olist*.

3.3 ANÁLISE DOS RESULTADOS

O *output* dos algoritmos indica a que *cluster* pertence cada cliente. Por exemplo, ao final do processo, o cliente A pertencerá ao *cluster* 1, enquanto o cliente B pertencerá ao *cluster* 2. Porém, apenas com essas informações, não é possível saber qual dos *clusters* performa melhor ou, ainda, qual deles possui os clientes menos rentáveis. Ou seja, os algoritmos não retornam uma caracterização detalhada de cada *cluster*.

Para se obter isso, é necessário analisar cada *cluster* individualmente com algumas métricas estatísticas (como média, mediana, máximos e mínimos), sob ótica das três variáveis RFV. Por exemplo, em um determinado *cluster*, a recência média é de x dias, os clientes compram, em média, y itens e o gasto médio é de z reais. Dentro da linguagem *Python* existem funções e métodos capazes de fornecer essas informações, tais como a função *describe().* Outra análise importante no que tange à caracterização dos grupos, é entender a distribuição dos clientes em cada um deles. Além disso, saber o quanto determinado *cluster* contribui para o faturamento da empresa, dentre outros indicadores.

Após a caracterização de cada *cluster*, é possível compará-los entre si e entender qual dos grupos performa melhor em relação ao valor gasto, por exemplo, ou qual grupo é mais fiel quando se olha para a recência e retorna mais vezes à *Olist*. Dessa análise, fica claro à empresa qual é o grupo de clientes que é mais rentável para ela. Do contrário, caso a empresa queira dedicar esforços para fidelizar clientes menos rentáveis, também é capaz de encontrá-los depois desta análise.

Vale ressaltar que o modelo, na forma como é construído, não tem a função de ranquear os grupos encontrados ‘do melhor para o pior’. Essa classificação é relativa e depende do objeto da empresa com a clusterização RFV. Por vezes é possível encontrar um cliente fiel à empresa (recência curta), mas que gasta um valor muito baixo, quando comparado com um cliente que compra uma vez ao ano, mas gasta um alto valor.

Por fim, serão criadas ações para cada um dos grupos encontrados, com base no perfil deles. Estas ações não serão aplicadas, mas ficarão como sugestões. Os diferentes perfis de clientes exigem diferentes ações, que vão desde implementar um programa de fidelidade a um programa de descontos massivos.

4 RESULTADOS

A corrente seção tem por objetivo comparar os resultados gerados pelos algoritmos *k-means* e AGNES, a partir da aplicação da metodologia, detalhada na seção anterior. Primeiramente, são apresentadas como se deram a seleção dos dados e construção das variáveis do modelo RFV. Após, tem-se a aplicação dos dois algoritmos para o conjunto de dados. Por fim, os algoritmos são comparados por meio do Coeficiente de Silhueta e ações são propostas para os grupos de clientes encontrados pelo método com melhor desempenho.

Conforme já mencionado, a linguagem de programação utilizada é o *Python*, na ferramenta *Jupyter Notebook*. Por convenção, a primeira etapa é importar os pacotes e bibliotecas que serão utilizadas ao longo do código, ou seja, as primeiras linhas de programação são destinadas às importações (FIGURA 15).

Figura 15 - Importação dos pacotes necessários

```
In [1]: # importando os pacotes necessários

        # para análise de dados
        import pandas as pd

        # para formatação de variáveis 'data-hora'
        import datetime as dt

        # para álgebra linear
        import numpy as np

        # para plotagem de gráficos
        import matplotlib.pyplot as plt

        # para transformação e normalização de dados
        from sklearn import preprocessing

        # algoritmo de clusterização k-means
        from sklearn.cluster import KMeans

        # algoritmo de clusterização AGNES
        from sklearn.cluster import AgglomerativeClustering

        # para gerar o dendrograma
        from scipy.cluster.hierarchy import dendrogram, linkage

        # para gerar o Coeficiente Silhueta
        from sklearn import metrics
```

FONTE: O autor (2020).

4.1 SELEÇÃO DOS DADOS

Neste projeto foram utilizados 3 conjuntos de dados:

- *olist_customers_dataset.csv* (com dados dos clientes);
- *olist_orders_dataset.csv* (com dados dos pedidos);
- *olist_order_payments_dataset.csv* (com dados dos pagamentos).

Por meio de três passos, as bases de dados foram carregadas e armazenadas em um único *dataframe* (*df*), uma estrutura tabular de n-dimensões. O primeiro deles foi a criação de uma variável que recebeu cada um dos arquivos no formato *.csv* (FIGURA 16).

Figura 16 – Criação das variáveis que receberam os arquivos no formato *.csv*

```
In [2]: # carregando os conjuntos de dados

        # base com as informações dos clientes
        file_cliente = 'olist_customers_dataset.csv'

        # base com as informações dos pedidos
        file_pedido = 'olist_orders_dataset.csv'

        # base com as informações do pagamento dos pedidos
        file_pagamento = 'olist_order_payments_dataset.csv'
```

FONTE: O autor (2020).

No segundo passo, com o auxílio do método *read_csv(),* cada um dos arquivos foi lido e atribuído a um *df* separado. Vale lembrar que o método *read_csv()* foi utilizado neste caso porque os arquivos foram extraídos do *Kaggle* no formato *.csv.* Se estivessem salvos no formato padrão do *Excel* (*xlsx)*, por exemplo, deveria ser utilizado o método *read_excel()*.

Ainda nesta etapa, é utilizada a função *head()*, que retorna as cinco primeiras linhas do *df*. O objetivo é conferir se a leitura do *df* foi feita corretamente. Da mesma maneira, poderia ser utilizada a função *tail()* para ver as últimas cinco linhas do *df*, em vez das cinco primeiras.

A Figura 17 representa o *df* criado a partir da base de clientes. Algumas colunas foram renomeadas para tornar o *df* mais legível dentro deste trabalho. A Figura 18, por sua vez, representa o *df* formado a partir da base de pedidos. Neste *df*, criou-se uma coluna auxiliar, com a data e hora da aprovação do pedido no formato desejado, de modo a facilitar os cálculos futuros.

Figura 17 - *Dataframe* de clientes

```
In [3]: # conjunto de dados dos clientes
df_cliente = pd.read_csv(file_cliente, sep= ';')

# este método renomeia as colunas
df_cliente.rename(
          columns={'customer_zip_code_prefix': 'customer_prefix_cep',
                   'customer_state': 'customer_uf'}, inplace=True)

# este método é utilizado para visualização das 5 primeiras linhas do df
df_cliente.head()
```

Out[3]:

	customer_id	customer_unique_id	customer_prefix_cep	customer_city	customer_uf
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruzes	SP
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP

FONTE: O autor (2020).

Figura 18 - *Dataframe* de pedidos

```
In [4]: # conjunto de dados dos pedidos
df_pedido = pd.read_csv(file_pedido, sep= ',')

# transformando o tipo da variável no formato 'data-hora'
df_pedido['order_approved_at2'] = pd.to_datetime(df_pedido['order_approved_at'])

# deletando a coluna no formato incorreto
df_pedido = df_pedido.drop(['order_approved_at'], axis=1)

# este método é utilizado para visualização das 5 primeiras linhas do df
df_pedido.head()
```

Out[4]:

	order_id	customer_id	order_status	order_purchase_timestamp
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39

order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	order_approved_at2
2017-10-04 19:55:00	2017-10-10 21:25:13	2017-10-18 00:00:00	2017-10-02 11:07:15
2018-07-26 14:31:00	2018-08-07 15:27:45	2018-08-13 00:00:00	2018-07-26 03:24:27
2018-08-08 13:50:00	2018-08-17 18:06:29	2018-09-04 00:00:00	2018-08-08 08:55:23
2017-11-22 13:39:59	2017-12-02 00:28:42	2017-12-15 00:00:00	2017-11-18 19:45:59
2018-02-14 19:46:34	2018-02-16 18:17:02	2018-02-26 00:00:00	2018-02-13 22:20:29

FONTE: O autor (2020).

A Figura 19 representa o *df* criado a partir da base de pagamento dos pedidos. Nesta base não foi feita nenhuma modificação.

Figura 19 - *Dataframe* de pagamentos

```
In [5]: # conjunto de dados do pagamento dos pedidos
        df_pagamento = pd.read_csv(file_pagamento, sep= ';')

        # este método é utilizado para visualização das 5 primeiras linhas do df
        df_pagamento.head()
```

Out[5]:

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45

FONTE: O autor (2020).

O terceiro e último passo é a junção desses três *dataframes* em um único *df*. Para esse cruzamento, utiliza-se o método *merge().* As bases de clientes e pedidos são unidas em uma nova base pela chave '*customer_id'*. Esta nova base, por sua vez, é unida ao *df* de pagamento pela chave '*order_id*' (FIGURA 20).

Figura 20 - Criação de um *df* único

```
In [6]: # cruzamento das bases
        df = pd.merge(df_cliente, df_pedido, on ='customer_id')
        df = pd.merge(df,df_pagamento, on = 'order_id')
```

FONTE: O autor (2020).

4.1.1 Tratamento dos dados

Para garantir dados corretos para os cálculos futuros, foram excluídos os valores nulos deste *df* criado. Alternativamente, estes valores poderiam ser substituídos por zeros. Neste trabalho, eles foram apenas deletados.

A Figura 21 mostra a execução do método *info()*, utilizado para identificar os tipos de cada variável, valores não nulos e uso de memória. Pode-se notar que as colunas de número 8, 9 e 11 possuíam valores nulos, visto que a quantidade de valores não-nulos é diferente de 103.886, a quantidade de linhas. Estes valores foram excluídos com o auxílio da função *notnull()*, mostrado na Figura 22. Ainda na mesma figura, para conferir se a exclusão foi feita corretamente, há a soma da quantidade de valores nulos em cada coluna que, após a aplicação da função *notnull()*, deve ser zero.

Figura 21 – Aplicação do método *info()*

```
In [7]: # este método é utilizado para identificar:
            # os tipos de cada variável,
            # valores não nulos e
            # uso de memória

        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103886 entries, 0 to 103885
Data columns (total 16 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   customer_id                    103886 non-null  object
 1   customer_unique_id             103886 non-null  object
 2   customer_prefix_cep            103886 non-null  int64
 3   customer_city                  103886 non-null  object
 4   customer_uf                    103886 non-null  object
 5   order_id                       103886 non-null  object
 6   order_status                   103886 non-null  object
 7   order_purchase_timestamp       103886 non-null  object
 8   order_delivered_carrier_date   101998 non-null  object
 9   order_delivered_customer_date  100754 non-null  object
 10  order_estimated_delivery_date  103886 non-null  object
 11  order_approved_at2             103711 non-null  datetime64[ns]
 12  payment_sequential             103886 non-null  int64
 13  payment_type                   103886 non-null  object
 14  payment_installments           103886 non-null  int64
 15  payment_value                  103886 non-null  float64
dtypes: datetime64[ns](1), float64(1), int64(3), object(11)
memory usage: 13.5+ MB
```

FONTE: O autor (2020).

Figura 22 - Exclusão dos valores nulos via função *notnull*()

```
In [8]: # este método deleta os valores nulos das colunas selecionadas
        df = df[pd.notnull(df['order_delivered_carrier_date'])]
        df = df[pd.notnull(df['order_delivered_customer_date'])]
        df = df[pd.notnull(df['order_approved_at2'])]

        # este método soma a quantidade de valores nulos por coluna
        df.isnull().sum(axis=0)
```

```
Out[8]: customer_id                    0
        customer_unique_id             0
        customer_prefix_cep            0
        customer_city                  0
        customer_uf                    0
        order_id                       0
        order_status                   0
        order_purchase_timestamp       0
        order_delivered_carrier_date   0
        order_delivered_customer_date  0
        order_estimated_delivery_date  0
        order_approved_at2             0
        payment_sequential             0
        payment_type                   0
        payment_installments           0
        payment_value                  0
        dtype: int64
```

FONTE: O autor (2020).

4.1.2 Criação das variáveis RFV

A partir do *df* criado anteriormente, foram calculadas as variáveis de recência, frequência e valor, utilizando como referência os seguintes conceitos:

- **Recência**: intervalo em dias entre 01/10/2018 e data da compra mais recente feita pelo cliente;
- **Frequência**: quantidade total de itens comprados a partir da primeira compra até 01/10/2018;
- **Valor**: valor total gasto por cada cliente a partir da primeira compra até 01/10/2018.

Foi escolhida 01/10/2018 como data de referência, pois os pedidos analisados foram realizados entre setembro/2016 e setembro/2018. Assim, qualquer data a partir de outubro/2018 poderia ser escolhida. Utilizou-se a função *datetime()* para converter a data 01/10/2018 no formato desejado.

Na dimensão *recência*, utilizou-se a função *max(),* que retorna a data da compra mais recente feita pelo cliente. Assim, se o cliente fez apenas uma compra, a data desta será retornada. Na hipótese de o cliente ter feito mais de uma, a data mais recente dentre as compras será retornada. Na dimensão *frequência*, utilizou-se a função *len(),* que calcula o comprimento de determinada variável. Neste caso, foi retornada a quantidade de itens comprados em uma ou mais compras. Na dimensão *valor*, utilizou-se a função *sum(),* que soma os valores gastos em cada compra. Os cálculos das variáveis RFV são feitos e agrupados para cada *customer_unique_id*, de modo que cada linha do *df* representa um único cliente (FIGURA 23).

Ao final desta etapa, tem-se um novo *df*, composto por quatro colunas (cliente, recência, frequência e valor). E é este *dataframe* utilizado nos algoritmos de clusterização. Na Figura 24 estão expressos dois métodos utilizados para uma primeira análise deste *dataframe*. Com o método *info()*, já visto anteriormente, pode-se notar a ausência de valores nulos, bem como o tipo de cada variável. Com o método *describe()*, obtém-se um resumo estatístico descritivo, no qual pode-se observar a quantidade de clientes no *df*, além de se ter conhecimento da média, desvio padrão, mínimos, máximos e alguns percentis de cada uma das variáveis construídas. Fica claro a presença de *outliers*, que podem ser removidos ou não, a depender dos objetivos.

Figura 23 - Construção das variáveis RFV

In [9]:
```
# criação da variável 'data de referência' no formato desejado
dt_ref = dt.datetime(2018,10,1)


# criação das variáveis de recência, frequência e valor, agrupando as variáveis por cliente
df1 = df.groupby('customer_unique_id').agg({
    'order_approved_at2': lambda x: (dt_ref - x.max()).days,
    'order_id': lambda x: len(x),
    'payment_value': lambda x: x.sum()})


# tratamento do df (renomeando colunas)
df1.rename(columns={'order_approved_at2': 'Recência',
                    'order_id': 'Frequência',
                    'payment_value': 'Valor'}, inplace=True)

# este método é utilizado para visualização das 5 primeiras linhas do df
df1.head()
```

Out[9]:

customer_unique_id	Recência	Frequência	Valor
0000366f3b9a7992bf8c76cfdf3221e2	143	1	141.90
0000b849f77a49e4a4ce2b2a4ca5be3f	146	1	27.19
0000f46a3911fa3c0805444483337064	569	1	86.22
0000f6ccb0745a6a4b88665a16c9f078	353	1	43.62
0004aac84e0df4da2b147fca70cf8255	320	1	196.89

FONTE: O autor (2020).

Figura 24 - Análise do *dataframe*

In [10]:
```
# este método é utilizado para identificar os tipos de cada variável, valores nulos e uso de memória
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 93341 entries, 0000366f3b9a7992bf8c76cfdf3221e2 to ffffd2657e2aad2907e67c3e9daecbeb
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Recência    93341 non-null  int64  
 1   Frequência  93341 non-null  int64  
 2   Valor       93341 non-null  float64
dtypes: float64(1), int64(2)
memory usage: 2.8+ MB
```

In [11]:
```
# este método fornece um resumo estatístico descritivo do df
df1.describe()
```

Out[11]:

	Recência	Frequência	Valor
count	93341.000000	93341.000000	93341.000000
mean	268.946604	1.079258	165.197318
std	152.584562	0.449146	226.329624
min	32.000000	1.000000	9.590000
25%	145.000000	1.000000	63.050000
50%	250.000000	1.000000	107.780000
75%	377.000000	1.000000	182.540000
max	726.000000	33.000000	13664.080000

FONTE: O autor (2020).

No que se refere à capacidade de processamento e uso de memória, o algoritmo AGNES já parte em desvantagem (BOSCARIOLI, 2008). Enquanto o algoritmo *k-means* consome n *bytes* de memória, o AGNES consome n^2 *bytes.* Considerando que o *df* original possui 93.341 linhas e 4 colunas, e considerando ainda que o computador utilizado possui sistema operacional de 64 *bits* e memória RAM de 8 GB, seria possível rodar por completo apenas o algoritmo *k-means*. Assim, para tornar os dois algoritmos comparáveis, optou-se por rodar ambos para uma amostra aleatória de 10% do *df*, o máximo possível para que se consiga executar o algoritmo AGNES sem estourar a memória do computador.

Com o auxílio do método *sample()*, se extraiu uma amostra aleatória do *df* original. Foi utilizado o parâmetro *'frac = 0.1'* para selecionar 10% da população. Quanto maior esse parâmetro, maior o tamanho da amostra (FIGURA 25).

Figura 25 - Seleção de uma amostra aleatória

```
In [12]: # este método retorna uma amostra aleatória do df
         df1 = df1.sample(frac = 0.1)

In [13]: # este método fornece um resumo estatístico descritivo do df
         df1.describe()
```

Out[13]:

	Recência	Frequência	Valor
count	9334.000000	9334.000000	9334.000000
mean	270.869831	1.078530	166.032370
std	153.144524	0.443948	227.393876
min	33.000000	1.000000	12.280000
25%	146.000000	1.000000	63.100000
50%	251.500000	1.000000	108.265000
75%	381.000000	1.000000	184.835000
max	726.000000	19.000000	7274.880000

FONTE: O autor (2020).

Ainda na Figura 25, há um novo resumo estatístico descritivo, válido apenas para esta amostra, obtido por meio do método *describe()*. Quando comparado com o resumo descritivo de toda a população, pode-se notar que as médias e percentis não sofrem grandes variações. Ademais, para a população de 93.341 clientes, considerando um erro amostral de 2%, um nível de confiança de 99% e uma distribuição de população mais heterogênea (50/50), qualquer amostra com mais de 3.971 clientes pode ser utilizada (COMENTTO, 2018). Portanto, pode-se concluir que essa amostra com 9.334 clientes, apesar de aleatória, é representativa.

Como os algoritmos são executados de maneira independente, e para evitar quaisquer erros e sobreposições de dados, a amostra do *df* é armazenada individualmente para cada algoritmo, como mostra a Figura 26. A função *head()* é utilizada apenas para comprovar que se trata do mesmo *df*.

Figura 26 - Atribuição da amostra em variáveis específicas

```
In [14]: # atribuindo a amostra do df para modelagem do algoritmo k-means
         df_kmeans = df1
         df_kmeans.head()
```

Out[14]:

customer_unique_id	Recência	Frequência	Valor
2cafc13ea55ac44b4288351bc60b861f	455	1	115.45
f8baafbeeee68555e4ed9243728815ba	235	1	185.18
33f965a01c42a564872f94319a636b59	310	1	75.16
ee226c6ced4b9ec3a0beebbd59942d69	323	1	102.38
2043559e77298e5c92dccccd693424e9	118	1	178.90

```
In [15]: # atribuindo a amostra do df para modelagem do algoritmo AGNES
         df_agnes = df1
         df_agnes.head()
```

Out[15]:

customer_unique_id	Recência	Frequência	Valor
2cafc13ea55ac44b4288351bc60b861f	455	1	115.45
f8baafbeeee68555e4ed9243728815ba	235	1	185.18
33f965a01c42a564872f94319a636b59	310	1	75.16
ee226c6ced4b9ec3a0beebbd59942d69	323	1	102.38
2043559e77298e5c92dccccd693424e9	118	1	178.90

FONTE: O autor (2020).

4.2 MODELAGEM

Antes da execução dos algoritmos, os dados dos *df* foram normalizados. Esta etapa é importante para evitar que as diferentes dimensões das variáveis interfiram na clusterização. Para a normalização dos dados foi utilizado o método *fit_transform()* (FIGURA 27).

Figura 27 - Normalização dos dados

```
In [16]: # normalização dos dados
         scaler = preprocessing.StandardScaler()
         df_norm_kmeans = scaler.fit_transform(df_kmeans)
         df_norm_agnes = scaler.fit_transform(df_agnes)
```

FONTE: O autor (2020)

4.2.1 Algoritmo *k-means*

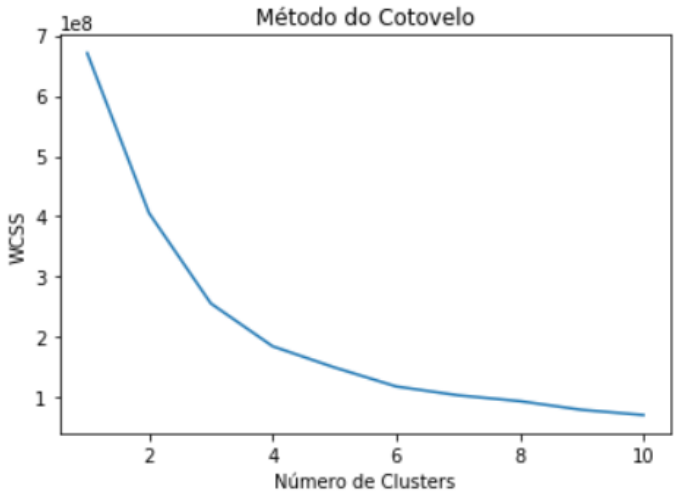
A Figura 28 representa a construção e execução do Método do Cotovelo, utilizado para definir o valor do parâmetro k (número de *clusters*). Primeiramente, cria-se uma lista vazia, que receberá os valores da inércia (*wcss)* do agrupamento. Após, há uma estrutura de repetição em forma de laço, a qual calcula o valor da inércia para cada k (número de *clusters*) de 1 a 10, e acrescenta à lista criada. Enfim, plota-se o gráfico, no qual o eixo horizontal representa o número de *clusters* e eixo vertical representa os valores *wcss*. Pela curva, pode-se concluir que $k = 4$, ou seja, a clusterização deve retornar quatro grupos. Vale ressaltar que a escolha do número de *clusters* é arbitrária. Apesar do Método do Cotovelo indicar uma boa opção para o número de *clusters*, a escolha deste parâmetro é de quem executa o algoritmo e pode ser qualquer outro número. Este trabalho seguiu com o sugerido pelo método ($k = 4$).

Figura 28 - Método do Cotovelo

```
In [17]: # criando uma lista para armazenar os valores wcss (inércia)
         wcss = []

         # um loop que relaciona a quantidade de clusters com a inércia
         for i in range(1, 11):
             kmeans = KMeans(n_clusters = i, init = 'random')
             kmeans.fit(df_kmeans)
             wcss.append(kmeans.inertia_)

         # plotagem do gráfico com o Método do Cotovelo
         plt.plot(range(1, 11), wcss)
         plt.title('Método do Cotovelo')
         plt.xlabel('Número de Clusters')
         plt.ylabel('WCSS') # within cluster sum of squares
         plt.show()
```



FONTE: O autor (2020).

A partir do método *KMeans(),* da biblioteca *scikit-learn*, e passando como parâmetros $k = 4$ e uma semente aleatória de inicialização, o algoritmo pôde ser executado com o auxílio do método *fit()*, conforme mostra a Figura 29. A semente neste caso é utilizada para gerar aleatoriamente números de inicialização dos centroides. Outra semente inicializa outros centroides e, assim, o resultado da clusterização pode ser diferente.

O método *fit()* cria as 'regras' de clusterização. O método *predict()* pode ser utilizado para aplicar essas regras criadas em novos dados, ou seja, clusteriza novos dados sem precisar rodar todo o *k-means* novamente, baseado nas regras criadas inicialmente.

Figura 29 - Algoritmo *k-means*

In [18]:
```
# aplicação do algoritmo
kmeans = KMeans(n_clusters = 4, random_state= 1810)
kmeans = kmeans.fit(df_norm_kmeans)

# cria coluna 'cluster' no df
df_kmeans['Cluster']= kmeans.predict(df_norm_kmeans)
df_kmeans
```

Out[18]:

customer_unique_id	Recência	Frequência	Valor	Cluster
2cafc13ea55ac44b4288351bc60b861f	455	1	115.45	0
f8baafbeeee68555e4ed9243728815ba	235	1	185.18	2
33f965a01c42a564872f94319a636b59	310	1	75.16	0
ee226c6ced4b9ec3a0beebbd59942d69	323	1	102.38	0
2043559e77298e5c92dccccd693424e9	118	1	178.90	2
...	...	...	...	...
f650b24f5ed49e64e9d088df058196b3	511	1	79.19	0
9073d24f5628a0ffdd7f9c686f7f3f5b	74	1	70.49	2
5cffd635d8749c43510edef193be34c8	71	1	118.03	2
73d1326ea19116c80b3a05fac829074e	347	1	99.22	0
fe7780b2659c715218834f592a838f6a	535	1	99.62	0

9334 rows × 4 columns

FONTE: O autor (2020).

Algumas medidas descritivas podem ser calculadas, a partir dos *clusters* gerados. Para cada grupo, tem-se a quantidade de clientes, a média e mediana de cada variável RFV, bem como os valores máximos e mínimos (FIGURA 30). É com este resumo que pode-se tomar conclusões acerca de cada *cluster*. Por exemplo, o *cluster* 3 é composto pelos clientes que mais gastam, além de ser o menor grupo. O *cluster* 0, por sua vez, além de ser o que menos gasta, é composto pelos clientes com a maior recência, ou seja, que compraram a mais tempo.

Figura 30 - Medidas descritivas dos resultados do algoritmo *k-means*

```
In [20]: # medidas descritivas
         medidas = ['count','min', 'mean', 'median', 'max']
         colunas = ['Recência','Frequência','Valor']

         agrup = df_kmeans.groupby(['Cluster'])

         resumo_kmeans = agrup[colunas].agg(medidas)
         resumo_kmeans
```

Out[20]:

	Recência					Frequência					Valor				
	count	min	mean	median	max	count	min	mean	median	max	count	min	mean	median	max
Cluster															
0	3551	293	424.456491	413	726	3551	1	1.000000	1	1	3551	13.78	133.280048	101.73	689.87
1	517	35	269.514507	256	633	517	2	2.276596	2	11	517	17.78	212.960600	153.61	1760.75
2	4973	32	160.869696	163	292	4973	1	1.000000	1	1	4973	14.78	133.127245	103.61	636.18
3	293	45	263.614334	258	605	293	1	1.068259	1	2	293	600.43	1084.184437	886.12	4163.51

FONTE: O autor (2020).

Por fim, foi calculado o Coeficiente de Silhueta para os resultados do algoritmo *k-means*. A função *silhouette_samples()* retorna o coeficiente para cada cliente enquanto a função *silhouette_score()* retorna o *score* geral da clusterização, calculado a partir do coeficiente médio dos grupos (FIGURA 31).

Figura 31 – Coeficiente de Silhueta dos resultados do algoritmo *k-means*

```
In [22]: # coeficiente silhueta para cada amostra
         cs_kmeans = metrics.silhouette_samples(df_kmeans, df_kmeans['Cluster'])
         cs_kmeans

Out[22]: array([0.53856334, 0.28811815, 0.1315512 , ..., 0.51064226, 0.36015312,
                0.51724774])

In [23]: # coeficiente silhueta para a clusterização k-means
         cs_kmeans_final = metrics.silhouette_score(df_kmeans, df_kmeans['Cluster'])
         cs_kmeans_final

Out[23]: 0.33296667539201386
```

FONTE: O autor (2020).

Para essa amostra do conjunto de dados da *Olist,* o Coeficiente de Silhueta é aproximadamente 0,333. De acordo com o que foi visto na Seção 2.4.2, valores entre 0 e 1 indicam um bom agrupamento dos dados. Esse é o coeficiente que será comparado com o encontrado pelo método AGNES.

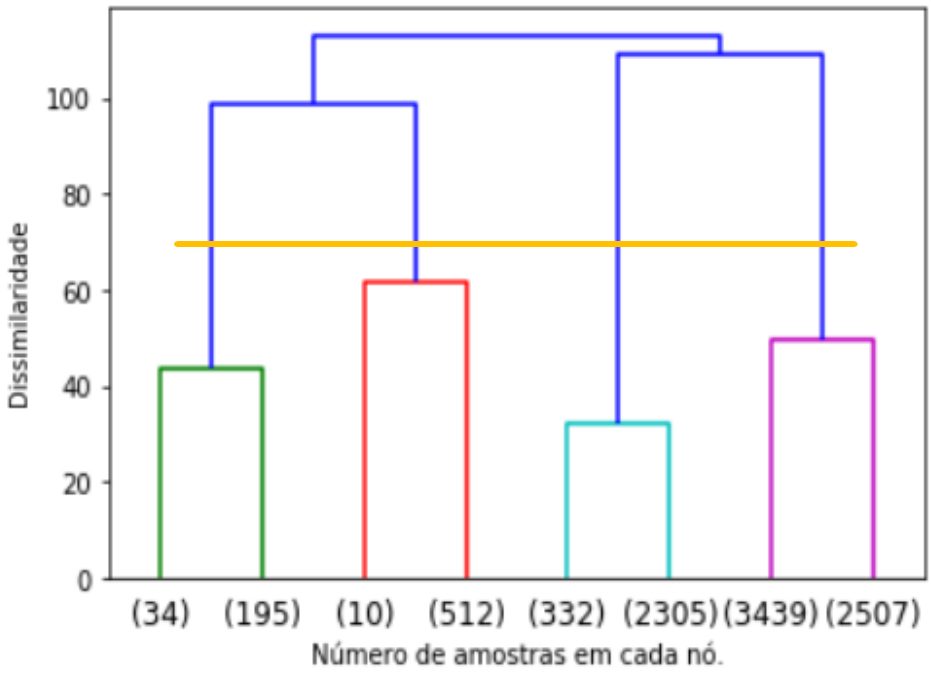
4.2.2 Algoritmo AGNES

Para o AGNES, não há um método convencional para se determinar o número ótimo de *clusters*, como o Método do Cotovelo para o *k-means*. Os algoritmos do tipo hierárquico geram um dendrograma e é a partir deste gráfico que se pode estimar o número ideal de *clusters*. A decisão sobre a quantidade de agrupamentos finais é chamada de corte no dendrograma. Esse corte pode ser feito em qualquer nó, a depender do nível de dissimilaridade entre *clusters* e a quantidade de grupos que se busca, e é representado por uma linha horizontal que corta todo o gráfico. A quantidade de ramos cortados no dendrograma é o número de *clusters*. O passo onde os valores de dissimilaridade mudam abruptamente podem indicar um bom ponto para definir o agrupamento final.

A Figura 32 representa o dendrograma obtido pelo algoritmo AGNES. Pode-se notar que, quanto maior a quantidade de grupos, menor a dissimilaridade (ou maior similaridade) entre os dados de um mesmo grupo. Para 3 grupos, a dissimilaridade é 100. Para 4 grupos, é pouco mais de 60. Para 5 grupos, próximo a 50. A queda mais abrupta na dissimilaridade ocorre na mudança de 3 para 4 *clusters*. Portanto, para a amostra analisada, foi escolhido um número de grupos igual a quatro, decisão representada pelo corte na cor amarela. Os grupos estão representados no dendrograma cada um por uma cor diferente: verde, vermelho, azul claro e roxo (FIGURA 32).

O código para construção e plotagem deste dendrograma está descrito na Figura 33. Dado que se trata de um algoritmo de clusterização hierárquico aglomerativo, foi utilizado o método *AgglomerativeClustering()*, para gerar os agrupamentos plotados no dendrograma.

Figura 32 – Dendrograma obtido pelo algoritmo AGNES



FONTE: O autor (2020).

Figura 33 - Dendrograma AGNES

In [19]:
```
# gráfico de visualização dos clusters - dendrograma

def plot_dendrogram(model, **kwargs):

    # cria a matriz de ligação
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # plota o dendrograma
    dendrogram(linkage_matrix, **kwargs)


# definição da matriz
agnes = AgglomerativeClustering(distance_threshold = 0, n_clusters = None, linkage = 'ward')
model = agnes.fit(df_norm_agnes)

# plotagem do dendrograma
plot_dendrogram(model, truncate_mode='level', p=2)
plt.ylabel('Dissimilaridade')
plt.xlabel("Número de amostras em cada nó.")
plt.show()
```

FONTE: O autor (2020).

Para execução e análise do algoritmo AGNES, novamente se utilizou o método *AgglomerativeClustering()*, da biblioteca *scikit-learn.* Ao todo, são passados três parâmetros: o número de *clusters*, equivalente a quatro neste trabalho; a *distance_threshold* que obrigatoriamente deve ser *none*, quando se especifica um número de *clusters*; e a *linkage* que, por padrão, é *ward*, mas foi deixado em destaque, já que este trabalho cita outras opções para este parâmetro.

Assim como no algoritmo *k-means,* o AGNES também é executado com o auxílio do método *fit()* e o *cluster* de cada cliente é adicionado ao *df* original, como uma nova coluna (FIGURA 34). As medidas descritivas para os grupos gerados pelo algoritmo AGNES estão calculadas na Figura 35. Os *scores* do Coeficiente de Silhueta, tanto de cada cliente, quanto o geral, estão na Figura 36. Todos esses resultados são obtidos pelos mesmos métodos utilizados no algoritmo *k-means*. Para fins de comparação, o valor do Coeficiente de Silhueta para a amostra do conjunto de dados selecionado é aproximadamente 0,316, menor do que o encontrado pelo *k-means* (FIGURA 36).

Figura 34 - Algoritmo AGNES

In [24]:
```
# aplicação do algoritmo
agnes = AgglomerativeClustering(distance_threshold = None, n_clusters = 4, linkage = 'ward')
agnes = agnes.fit(df_norm_agnes)

# cria coluna 'Cluster' no df
df_agnes['Cluster']=agnes.fit_predict(df_norm_agnes)
df_agnes
```

Out[24]:

customer_unique_id	Recência	Frequência	Valor	Cluster
2cafc13ea55ac44b4288351bc60b861f	455	1	115.45	2
f8baafbeeee68555e4ed9243728815ba	235	1	185.18	1
33f965a01c42a564872f94319a636b59	310	1	75.16	1
ee226c6ced4b9ec3a0beebbd59942d69	323	1	102.38	1
2043559e77298e5c92dccccd693424e9	118	1	178.90	1
...	...	...	...	...
f650b24f5ed49e64e9d088df058196b3	511	1	79.19	2
9073d24f5628a0ffdd7f9c686f7f3f5b	74	1	70.49	1
5cffd635d8749c43510edef193be34c8	71	1	118.03	1
73d1326ea19116c80b3a05fac829074e	347	1	99.22	2
fe7780b2659c715218834f592a838f6a	535	1	99.62	2

9334 rows × 4 columns

FONTE: O autor (2020).

Figura 35 - Medidas descritivas dos resultados do algoritmo AGNES

```
In [26]: # medidas descritivas
medidas = ['count','min', 'mean', 'median', 'max']
colunas = ['Recência','Frequência','Valor']

agrup = df_agnes.groupby(['Cluster'])

resumo_agnes = agrup[colunas].agg(medidas)
resumo_agnes
```

Out[26]:

	Recência					Frequência					Valor				
	count	min	mean	median	max	count	min	mean	median	max	count	min	mean	median	max
Cluster															
0	522	35	269.622605	256.5	633	522	2	2.268199	2	11	522	17.78	217.400096	159.35	1094.63
1	5946	32	186.345947	188.0	412	5946	1	1.000000	1	1	5946	14.78	138.469157	106.90	715.32
2	2637	325	463.458096	453.0	726	2637	1	1.000000	1	1	2637	13.78	134.070903	95.67	911.65
3	229	45	231.148472	191.0	605	229	1	1.078603	1	4	229	611.81	1190.919127	1006.44	4163.51

FONTE: O autor (2020).

Figura 36 – Coeficiente de Silhueta dos resultados do algoritmo AGNES

```
In [28]: # coeficiente silhueta para cada amostra
cs_agnes = metrics.silhouette_samples(df_agnes, df_agnes['Cluster'])
cs_agnes

Out[28]: array([0.5826584 , 0.29888269, 0.09579257, ..., 0.43283986, 0.20295832,
       0.59459053])

In [29]: # coeficiente silhueta para a clusterização AGNES
cs_agnes_final = metrics.silhouette_score(df_agnes, df_agnes['Cluster'])
cs_agnes_final

Out[29]: 0.31615070305525333
```

FONTE: O autor (2020).

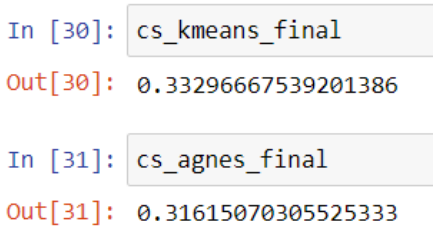
4.3 ANÁLISE DOS RESULTADOS

Esta seção tem por objetivo analisar o desempenho dos dois algoritmos executados, comparando seus respectivos coeficientes de silhueta e outras medidas de desempenho, como uso de memória computacional. Para os grupos gerados pelo algoritmo com melhor performance, serão sugeridas ações de rentabilização dos clientes do *e-commerce*.

4.3.1 Comparação dos algoritmos

Para a amostra analisada, a clusterização gerada pelo *k-means* possui maior Coeficiente de Silhueta, ou seja, os dados estão melhor agrupados por esse algoritmo (FIGURA 37).

Figura 37 - Comparação dos coeficientes de silhueta



FONTE: O autor (2020).

Para minimizar o impacto da aleatoriedade na seleção da amostra e tornar justa a escolha do melhor algoritmo, foram comparados os coeficientes de silhueta de outras dez amostras aleatórias (do mesmo conjunto), como mostra a Tabela 3.

Tabela 3 - Comparação dos coeficientes de silhueta para 10 amostras aleatórias

Número da amostra	**Número de *clusters***	**Coeficiente de Silhueta *k-means***	**Coeficiente de Silhueta AGNES**
1	4	0,41	0,25
2	3	0,42	0,30
3	4	0,32	0,33
4	4	0,34	0,18
5	4	0,42	0,24
6	3	0,43	0,35
7	3	0,29	0,36
8	4	0,32	0,30
9	4	0,33	0,35
10	4	0,32	0,30

FONTE: O autor (2020).

Nota-se que, mesmo com a aleatoriedade na seleção das amostras, o algoritmo *k-means* teve um Coeficiente de Silhueta maior em 70% dos casos, o que indica que este algoritmo performa melhor para o conjunto de dados selecionados no que tange à qualidade dos agrupamentos gerados.

Conforme supracitado, é possível executar o *k-means* para todo o conjunto de dados, sem a necessidade de retirar uma amostra para análise, pois a memória utilizada está dentro do limite suportado pelo computador. Esse atributo, aliado ao desempenho na clusterização medida pelo Coeficiente de Silhueta, tornam o algoritmo *k-means* melhor que o AGNES para este conjunto de dados da *Olist*.

Por fim, o Coeficiente de Silhueta dos resultados do algoritmo *k-means* aplicado em toda a base de clientes, não apenas em uma amostra, é aproximadamente 0,31 (FIGURA 38).

Figura 38 – Coeficiente de silhueta dos resultados do algoritmo *k-means* para a base toda

```
In [27]: # coeficiente silhueta para a clusterização k-means
         cs_kmeans_final = metrics.silhouette_score(df_kmeans, df_kmeans['Cluster'])
         cs_kmeans_final

Out[27]: 0.3147248584689809
```

FONTE: O autor (2020).

4.3.2 Caracterização dos grupos

Para a amostra do conjunto de dados da *Olist* analisada, foram encontrados quatro grupos pelo algoritmo *k-means*, cujas características estão dispostas na Tabela 4.

Tabela 4 - Resumo por *cluster*

Cluster	**Quantidade de clientes**	**Recência média (em dias)**	**Frequência média (em quantidade de itens)**	**Valor médio (em R$)**
0	3551	424	1	133,28
1	517	270	2	212,96
2	4973	161	1	133,12
3	293	264	1	1084,19

FONTE: O autor (2020).

As ações propostas, como mostra a Tabela 5, ficam apenas como sugestões, sem aplicação prática ou medição de sua efetividade.

Tabela 5 - Ações sugeridas por *cluster*

Cluster	**Ações sugeridas**
0	Descontos agressivos
1	Ofertar lançamentos Ofertar combos de produtos
2	Programa de fidelidade Condições diferenciadas de pagamento
3	Ofertar produtos mais caros Ofertar brindes

FONTE: O autor (2020).

O *cluster* 0, composto por 38% dos clientes, é o segundo maior grupo. É composto pelos clientes que, em média, compraram há mais tempo, 424 dias. A compra foi de apenas um item e de baixo valor. Devido a essa performance, o ideal é uma ação de descontos agressivos. Uma política de preços mais baratos para determinados produtos, deve chamar a atenção de alguns dos clientes, que retornarão ao *e-commerce* atraídos pelas promoções. A frequência e o valor podem não sofrer grandes alterações, visto que se trata de produtos mais baratos, mas a recência pode diminuir bastante. Como não é um grupo rentável, a comunicação deve ser feita pelos meios com menor custo.

O *cluster* 1, apesar de ser o grupo que comprou a maior quantidade de itens, possui uma das maiores recências, de 270 dias. O valor total médio gasto por esse grupo é maior do que de outros dois grupos, totalizando quase R$ 213,00. Porém, quando se analisa o valor gasto por item, este tende a ser menor do que os demais, já que sua frequência é de dois itens. Para explorar o potencial deste grupo, são propostas duas ações. A primeira delas, é ofertar lançamentos e produtos novos, com condições diferenciadas. A comunicação deverá ser voltada especialmente para estes clientes. A ideia é fomentar o acesso ao *e-commerce*, pensando em reduzir a alta recência do grupo. A segunda ação que pode ser feita, é utilizar a inteligência de dados para ofertar *kits* e produtos relacionados, visto que estes clientes estão dispostos a comprar mais de um item. Por exemplo, ofertar um celular e acessórios, como película para tela e capa protetora. Assim, se consegue aumentar também as variáveis frequência e valor dos clientes do grupo.

O *cluster* 2 é o maior grupo: reúne 53% dos clientes. É o *cluster* que, em média, comprou mais recentemente, 161 dias. Apesar dessa boa recência, o valor médio gasto pelos clientes é o menor entre os grupos, de R$ 133,00 em um único produto do *e-commerce.* Este é um grupo que, se não receber ações, pode se transformar no *cluster* 0, visto que ambos possuem frequência e valor parecidos e o que os difere é a recência. Deve-se, portanto, aproveitar esta recência curta para retê-los, antes que o relacionamento criado se perca. Por compor mais da metade da base de clientes, suas ações devem ser priorizadas. A sugestão é estruturar um programa de fidelidade para aumentar o engajamento desses clientes, de modo que o relacionamento com o *e-commerce* se mantenha ativo. Outra ação que poderia ser executada de forma paralela, é ofertar condições diferenciadas de pagamento. Isso inclui uma porcentagem maior de desconto à vista, quantidade de parcelas no boleto

ou cartão, etc. O objetivo é manter a curta recência e aumentar frequência e valor, tornando os clientes desse grupo promotores do *e-commerce*.

O *cluster* 3 é o menor grupo e possui os clientes que mais gastaram. Em média, somaram mais de R$ 1.000,00 em apenas um item, o que indica a compra de poucos produtos, mas de um valor bastante elevado. Esse grupo de clientes gastou cinco vezes mais do que o segundo grupo que mais gastou e até oito vezes mais do que os demais grupos. Seu bom desempenho, combinado com sua recência intermediária, de 264 dias, tornam o *cluster* 3 muito importante para a empresa. Embora seja composto pela menor porcentagem de clientes, gera retorno financeiro. Para este grupo, devem ser ofertados produtos mais caros, pois seus clientes parecem estar dispostos a gastar mais. Assim, seria possível reduzir a recência e aumentar o valor. Por fim, como recompensa pelo valor gasto, poderiam ser dados brindes ou mimos, quando o cliente comprar um produto mais caro.

CONSIDERAÇÕES FINAIS

Este trabalho teve por objetivo propor o uso de dois algoritmos de clusterização, baseado nas variáveis recência, frequência e valor, que compõem o modelo RFV, muito utilizado como uma ferramenta de CRM. Para entender como essas variáveis se relacionam entre si na formação de grupos, foram selecionados o método de clusterização por particionamento *k-means*, e o método de clusterização hierárquico *Agglomerative Nesting* (AGNES).

Como *input* da clusterização, foram escolhidos dados de clientes do *marketplace Olist*, cujas informações foram disponibilizadas no *Kaggle*, uma plataforma para hospedagem de bases de dados para projetos.

Após a aplicação dos dois algoritmos de clusterização, eles puderam ser comparados por meio do Coeficiente de Silhueta, um indicador que mede o quão bem as amostras estão agrupadas. Por esse coeficiente, pôde-se concluir que o *k-means* teve desempenho superior na clusterização do conjunto de dados escolhido. Das dez amostras avaliadas, sete delas tiveram Coeficiente de Silhueta maior pelo algoritmo *k-means*. Ademais, o consumo de memória para executar o AGNES para todo o conjunto de dados é consideravelmente maior do que para executar o *k-means*. Por essa razão, no computador utilizado foi possível clusterizar todo o conjunto apenas pelo *k*-means, o que o torna o melhor dentre os dois métodos.

Em um cenário onde o *e-commerce* vem ganhando cada vez mais relevância, entender o perfil de seu consumidor para traçar estratégias mais assertivas serve como vantagem competitiva para qualquer empresa frente à concorrência. Por isso, a partir dos dados da *Olist*, foram gerados quatro grupos, que foram caracterizados segundo o valor das variáveis RFV. De acordo com a performance de cada *cluster*, foram sugeridas ações de retenção e rentabilização dos clientes de cada grupo.

Por fim, visto que as ações não foram testadas e validadas, observa-se uma oportunidade para trabalhos futuros, na qual poderia ser explorada a efetividade das ações, o retorno financeiro de cada grupo e se os grupos mantêm o mesmo comportamento de compra ao longo de um determinado período.

REFERÊNCIAS

ARTHUR, D; VASSILVITSKII, SERGEI. **K-Means++: The Advantages of Careful Seeding.** Conference: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007. Disponível em: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf. Acesso em 03 de maio de 2020.

BAEZA-YATES, R. A; FRAKES, W. B. **Introduction to data structures and algorithms related to information retrieval**. Prentice-Hall, Inc., San Diego, CA, USA, p.13 – 27, 1992.

BIONDI NETO, L. et al. Minicurso de sistema especialista nebuloso. In: XXXVII Simpósio Brasileiro de Pesquisa Operacional. Goiânia: [s.n.], 2006.

BOSCARIOLI, CLODIS. **Análise de Agrupamentos baseada na Topologia dos Dados e em Mapas Auto-organizáveis.** 2008. 118 p. Tese (Doutorado em Engenharia Elétrica) – Departamento de Engenharia de Sistemas Eletrônicos, Escola Politécnica da Universidade de São Paulo – USP.
Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3142/tde-11082008-132720/publico/Tese_Final_Boscarioli.pdf. Acesso em: 23 de abril de 2020.

CARLANTONIO, L. M. di. Novas Metodologias para Clusterização de Dados. Tese (Doutorado) – Universidade Federal do Rio de Janeiro, 2001.

CASSIANO, K. **Análise de Séries Temporais Usando Análise Espectral Singular (SSA) e Clusterização de Suas Componentes Baseada em Densidade.** 2014. Tese (Doutorado em Engenharia Elétrica) – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio, Disponível em: https://www.maxwell.vrac.puc-rio.br/24787/24787_1.PDF. Acesso em: 23 de abril de 2020.

COMENTTO. Calculadora Amostral, 2018. Disponível em: https://comentto.com/calculadora-amostral/. Acesso em: 26 de agosto de 2020.

DANTAS, M; ROSA, F. **Utilização e Análise de Técnicas Alternativas para Elaboração de Segmentos RFV**. In: I EMA – ENCONTRO DE MARKETING DA ANPAD, Porto Alegre: Escola de Administração e PPGA da UFRGS, 2004.

EPOCA NEGÓCIOS. Disponível em: https://epocanegocios.globo.com/mercado/noticia/2020/05/epoca-negocios-com-covid-19-e-commerce-ja-e-48-maior-que-no-mesmo-periodo-de-2019.html. Acesso em: 04 jul. 2020.

EXAME. Disponível em: https://exame.com/economia/compras-pela-internet-disparam-ate-40-com-impacto-do-novo-coronavirus/. Acesso em: 04 jul. 2020.

FECOMERCIO. Disponível em: https://www.fecomercio.com.br/noticia/maior-acesso-a-internet-muda-comportamento-do-consumidor-online. Acesso em: 04 jul. 2020.

HONDA, H. **Introdução Básica à Clusterização**. Laboratório de Aprendizado de Máquina em Finanças e Organizações – LAMFO. 2017. Disponível em: https://lamfo-unb.github.io/2017/10/05/Introducao_basica_a_clusterizacao/. Acesso em 03 de maio de 2020.

KASSAMBARA, A. **Cluster Validation Statistics: Must Know Methods**. 2018. Não paginado. Disponível em: https://www.datanovia.com/en/lessons/cluster-validation-statistics-must-know-methods/#silhouette-coefficient. Acesso em: 17 de maio de 2020

KLEINA, M. **Identificação, Monitoramento e Previsão de Tempestades Elétricas.** 2015. 114 f. Tese (Doutorado em Ciências) – Departamento de Matemática, Setor de Ciências Exatas e Departamento de Construção Civil, Setor de Tecnologia, Universidade Federal do Paraná – UFPR, Curitiba, 2015. Disponível em: https://acervodigital.ufpr.br/handle/1884/41485. Acesso em: 23 de abril de 2020.

KODINARIYA. T. *Review on determining number of Cluster in K-Means Clustering.* **International Journal of Advance Reserach in Computer Science and Management Studies.** Rajkot, India, v.1, n. 6, p. 90-95, nov. 2013.

MEIO E MENSAGEM. 2020 Disponível em: https://www.meioemensagem.com.br/home/marketing/2020/02/21/e-commerce-cresce-alem-do-esperado-aponta-ebit-nielsen.html. Acesso em: 04 jul. 2020.

NIKUMANESH, E; ALBADVI, A. Customer's life-time value using the RFM model in the banking industry: a case study. **International Journal of Electronic Customer Relationship Management**, Tehran, Irã, v. 8, n. 1/2/3, p.15-30, jan. 2014.

NIMBALKAR, D. Data Mining using RFM Analysis. **International Journal of Scientific & Engineering Research**, v. 4, n. 12, dez. 2013.

NOGUEIRA, F. **Machine Learning aplicado a dados de e-commerce**. No prelo.

OLIST. **Conjunto de dados públicos de comércio eletrônico brasileiro por Olist.** 2018. Disponível em: https://www.kaggle.com/olistbr/brazilian-ecommerce. Acesso em: 28 de junho de 2020.

OLIVEIRA, T. **Clusterização de dados utilizando técnicas de redes complexas e computação bioinspirada**. 2008. 112 f. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação – ICMC/USP, Universidade de São Paulo, São Carlos (SP), 2008. Disponível em: https://teses.usp.br/teses/disponiveis/55/55134/tde-01042008-142253/pt-br.php. Acesso em: 23 de abril de 2020.

PEREIRA, G. R. J. **Um estudo sobre alguns métodos hierárquicos para análise de agrupamentos.** 1993. 155 f. Dissertação (Mestrado em Estatística) – Instituto de Matemática, Estatística e Ciência da Computação da Universidade Estadual de Campinas – UNICAMP, Campinas, São Paulo, 1993. Disponível em: http://repositorio.unicamp.br/jspui/handle/REPOSIP/307143. Acesso em: 14 de maio de 2020.

PINHO, A. Análise RFV do cliente por algoritmos genéticos na otimização de estratégias de marketing. **Revista Pensamento Contemporâneo em Administração**, Rio de Janeiro, v. 3, n. 2, p. 86-98, mai./ago. 2009.

PUTLER. **RFM Analysis for Successful Customer Segmentation**. 2017. Não paginado. Disponível em: https://www.putler.com/rfm-analysis/#resources. Acesso em: 22 de abril de 2020.

ROCHA, H. V; LACHI, R. L. **Aspectos Básicos de Clustering: Conceitos e Técnicas.** Relatório Técnico IC-05-003, UNICAMP – Universidade Estadual de Campinas, Campinas, fevereiro, 2005.

SANTANA, F. **Entenda o Algoritmo K-Means e Saiba como Aplicar essa Técnica.** Minerando Dados. 2017. Disponível em: https://minerandodados.com.br/entenda-o-algoritmo-k-means/. Acesso em: 03 de maio de 2020.

SAMPAIO, P. Entendendo k-means, agrupando dados e tirando camisas. **Blog Medium**. 2017, não paginado. Disponível em: https://medium.com/@paulo_sampaio/entendendo-k-means-agrupando-dados-e-tirando-camisas-e90ae3157c17. Acesso em 03 de maio de 2020.

SEIDEL, J. E; JUNIOR, M. J. F; ANSUJ, P. A.; NOAL, C. R. M. **Comparação entre o Método Ward e o método k-médias no agrupamento de produtores de leite.** Ciência e Natureza, Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, v. 30 (1), p. 7-15, 2008.

SCIKIT-LEARN (biblioteca do software de programação Python) https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html. Acesso em: 17 de maio de 2020.

TEMPORAL, J. Como definir o número de *clusters* para o seu *kmeans*. **Blog Pizza de Dados (Medium)**. 2017, não paginado. Disponível em: https://medium.com/pizzadedados/kmeans-e-metodo-do-cotovelo-94ded9fdf3a9. Acesso em 12 de maio de 2020.

TORRES, A, R. **Aplicação de métodos de clusterização em um estudo sobre o mercado acionário brasileiro.** 2013. 84 f. Dissertação (Mestrado em Matemática) – Departamento de Matemática, Centro Técnico da Pontifícia Universidade Católica – PUC-Rio, Rio de Janeiro, Rio de Janeiro. 2013. Disponível em: https://www.maxwell.vrac.puc-rio.br/22901/22901.PDF. Acesso em: 14 de maio de 2020.

VASCONCELOS, P. O que é RFM e como aplicá-lo ao seu time de Customer Success. **Blog MaxMilhas Tech**, 2017, não paginado. Disponível em: https://medium.com/maxmilhas-tech/o-que-%C3%A9-rfm-e-como-aplic%C3%A1-lo-ao-seu-time-de-customer-service-b9c35817ed01. Acesso em: 21 de abril de 2020.

WANG, F; FRANCO-PENYA, H-H; KELLEHER, D. J; PUGH, J; ROSS, R. **An Analysis of the Application of Simplified Silhouette to the Evaluation of k-means**

Clustering Validation. School of Computing, Dublin Institute of Technology, Ireland, jul. 2017.

WEI, J; LIN, S; WU, H. A review of the application of RFM model. **African Journal of Business Management**, Taiwan, v. 4, n. 19, p. 4199-4206, dez. 2010.