UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

Jhonata Querobim

Análise comparativa entre algoritmos de agrupamento e de detecção de comunidades em redes

São Carlos - SP
2021

Jhonata Querobim

Análise comparativa entre algoritmos de agrupamento e de detecção de comunidades em redes

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação da Universidade Federal de São Carlos, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientação Prof. Dr. Alan Demétrius Baria Valejo

São Carlos - SP
2021

Dedico este trabalho a toda minha rede de apoio que foi fonte de suporte em diversas etapas e situações da minha vida.

Agradecimentos

Primeiramente agradeço aos meus pais, Cláudia e Nicanor, que sempre me apoiaram para que eu fosse capaz de enfrentar todos os desafios que surgiram em minha vida. Eles foram os principais responsáveis pela minha formação acadêmica.

Agradeço ao professor Dr. Alan Valejo por ter me apresentado ao tema, pela orientação, oportunidade e paciência.

Agradeço meus amigos e colegas de turma pelo apoio, convivência e aprendizado, que compartilharam comigo parte deste tempo como aluno do curso de Bacharelado em Engenharia de Computação.

Na história da humanidade (e dos animais também) aqueles que aprenderam a colaborar e improvisar foram os que prevaleceram.
(Charles Darwin)

Resumo

O agrupamento é uma das técnicas de Aprendizado de Máquina mais notórias e com um grande número de aplicações práticas em diversas áreas de conhecimento. Entender os tipos de abordagens e suas características é de extrema importância para realizar um agrupamento bem sucedido. Nesse contexto, além dos algoritmos de agrupamento tradicionais, como o DBSCAN, uma abordagem que vem ganhando notoriedade na literatura são os algoritmos de agrupamento baseados em redes, que constroem uma rede a partir dos dados e utilizam detecção de comunidades para encontrar os grupos. Este estudo tem como objetivo realizar uma análise comparativa entre algoritmos de agrupamento e algoritmos de detecção de comunidades utilizando bases de dados geradas artificialmente. Além disso, foi realizada uma análise do impacto dos algoritmos de construção de redes no desempenho dos algoritmos de detecção de comunidade. Para os resultados foi calculada a Informação Mútua Normalizada (NMI), uma das principais métricas na avaliação de desempenho em problemas de agrupamento. Na análise dos resultados, foi estudado se, mesmo havendo diferença de desempenho, tal diferença representa ou não uma diferença estatisticamente significativa entre os algoritmos. Os resultados mostraram que os algoritmos de detecção de comunidades são uma alternativa viável e podem superar algoritmos tradicionais de agrupamento em alguns cenários.

Palavras-chave: Aprendizado de Máquina, Aprendizado não-supervisionado, Agrupamento, Detecção de Comunidades, Redes, Informação Mútua Normalizada.

Abstract

Clustering is one of the most notorious Machine Learning techniques and has an infinite number of practical applications in different areas of knowledge. Understanding the types of approaches and their characteristics is essential for successful clustering. In this context, in addition to traditional clustering algorithms such as DBSCAN, an approach that has gained notoriety in the literature is network-based clustering algorithms, which build a network from data and use communities to find groups. The purpose of this study is to carry out a comparative analysis between clustering algorithms and community detection algorithms in experiments performed with an artificially generated database. In addition, an analysis of the impact of graph generation algorithms on the performance of community detection algorithms was performed. For the results, the Normalized Mutual Information (NMI) was calculated, one of the main metrics in the performance evaluation in clustering problems. In the results of the experiments it was analyzed wheter, even with a difference in the algorithm performance, such difference represents or not a statistical difference. The results showed that community detection algorithms are a viable alternative and can outperform traditional clustering algorithms in some scenarios.

Keywords: Machine Learning, Unsupervised Learning, Clustering, Community Detection, Graphs, Mutual Normalized Information.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Com o avanço computacional e tecnológico o acesso à informação se expandiu de forma exponencial e, consequente a isso, a quantidade de informação gerada e armazenada em bases de dados. Nesse contexto, surge um grande desafio: extrair e processar informações relevantes nessa imensa quantidade de dados coletados. Dentre as principais áreas de pesquisa que lidam com esse desafio, está o Aprendizado de Máquina.

A palavra aprender é originada do latim e é composta por "ad" (junto) e "prae" (à frente) + "hendere" (relacionado a hera, uma planta trepadeira que se prende as paredes para poder crescer) [18]. O processo da aprendizagem é resultado de interações entre estruturas mentais e o meio ambiente. Segundo Skinner (1950), "aprendizagem é uma mudança na probabilidade de uma resposta específica" [39]. Este conceito pode ser aplicado para seres vivos e também para máquinas, assim como seres vivos respondem a estímulos do meio, um dispositivo artificial pode ser capaz de retornar respostas a entrada de dados.

No contexto de computação, o aprendizado de máquina, subárea da inteligência artificial, nos ajuda a identificar padrões, fazer previsões e tomar decisões com o mínimo possível de intervenção humana. Essa área tem como objetivo estudar e construir algoritmos que permitam aos computadores aprender e melhorar o seu desempenho através de experiência.

Os algoritmos de aprendizado de máquina podem ser divididos, de forma mais generalizada, em duas categorias: supervisionados e não supervisionados. No primeiro caso, considera-se dados rotulados para induzir um modelo e classificar novos exemplos. Na segunda categoria, os algoritmos inferem uma divisão natural para os dados, com base na estrutura topológica e informações adicionais relacionadas aos dados. Nesse contexto, uma área de estudo específica do Aprendizado de Máquina não supervisionado é o agrupamento. O agrupamento de objetos de acordo com suas semelhanças é base de grande parte da ciência, em diversas áreas do conhecimento como psicologia, marketing, medicina e outros, uma vez que, organizar dados em grupos é um dos modos mais fundamentais que configura a compreensão e aprendizado [20].

Tradicionalmente, o agrupamento de dados consiste na identificação de grupos de objetos de acordo com algum critério de similaridade associado aos atributos dos dados. Por exemplo, é possível utilizar medidas de distância entre objetos, tais como, distância euclidiana ou cosseno. Algoritmos de agrupamento clássicos são o kMeans [1] e o DBSCan [11].

Nos últimos anos, técnicas de agrupamento de dados baseadas em redes vêm ganhando

notoriedade e produzindo resultados expressivos quando comparados com algoritmos tradicionais da literatura. Redes são estruturas relacionais definidas como "sistemas físicos, biológicos ou sociais caracterizados por um conjunto de entidades bem definidas que se integram e interagem dinamicamente entre si" [43]. Nas redes as entidades são descritas por vértices e suas relações por arestas. Nesse sentido, é possível representar diversas estruturas por meio de redes: redes de esgoto, redes de telefonia, redes de computadores, redes de contatos, entre outras.

No caso de algoritmos de agrupamento baseados em redes, primeiro o conjunto de dados é transformado em uma rede e, em seguida, um algoritmo de Detecção de Comunidade é utilizado para particionar os vértices da rede em comunidades. Por fim, as comunidades são projetadas como grupos no conjunto de dados original. Uma das vantagens dos algoritmos de detecção de comunidades é poder examinar, além dos atributos dos objetos no conjunto de dados, a estrutura topológica oriunda da transformação dos dados em uma rede, bem como poder utilizar todo o arcabouço teórico oriundo da Teoria das Redes Complexas. Além disso, é comum situações em que os dados naturalmente já são representados através de relações. A Figura 1 ilustra esse processo. A Figura 1(a) define o conjunto de dados original; na Figura 1(d) os dados originais foram transformados em uma redes; na Figura 1(b) um algoritmo de detecção de comunidades foi utilizado para encontrar as comunidades; na Figura 1(b) as comunidades foram projetadas como grupos nos dados originais.

Considerando que algoritmos de agrupamento baseados em redes necessitam da construção de uma rede a partir dos dados analisados, existem diversos algoritmos que podem ser utilizados, os quais possuem estratégias diferentes e produzem redes com características topológicas distintas.

Assim, faz-se necessária a existência de pesquisas no sentido de analisar os métodos de construção de redes e avaliar se existe uma diferença significativa na precisão dos algoritmos de detecção de comunidades em conjunto com diferentes algoritmos de construção de redes quando comparados aos algoritmos tradicionais de agrupamento, tais como o kMeans e DBSCan.

1.1 Objetivos

1.1.1 Geral

O objetivo geral deste trabalho é realizar uma análise comparativa entre algoritmos de agrupamento e algoritmos de detecção de comunidades em conjunto com métodos de construção de redes. A hipótese que espera-se confirmar é de que não existe diferença estatisticamente significativa no desempenho dos algoritmos selecionados, considerando

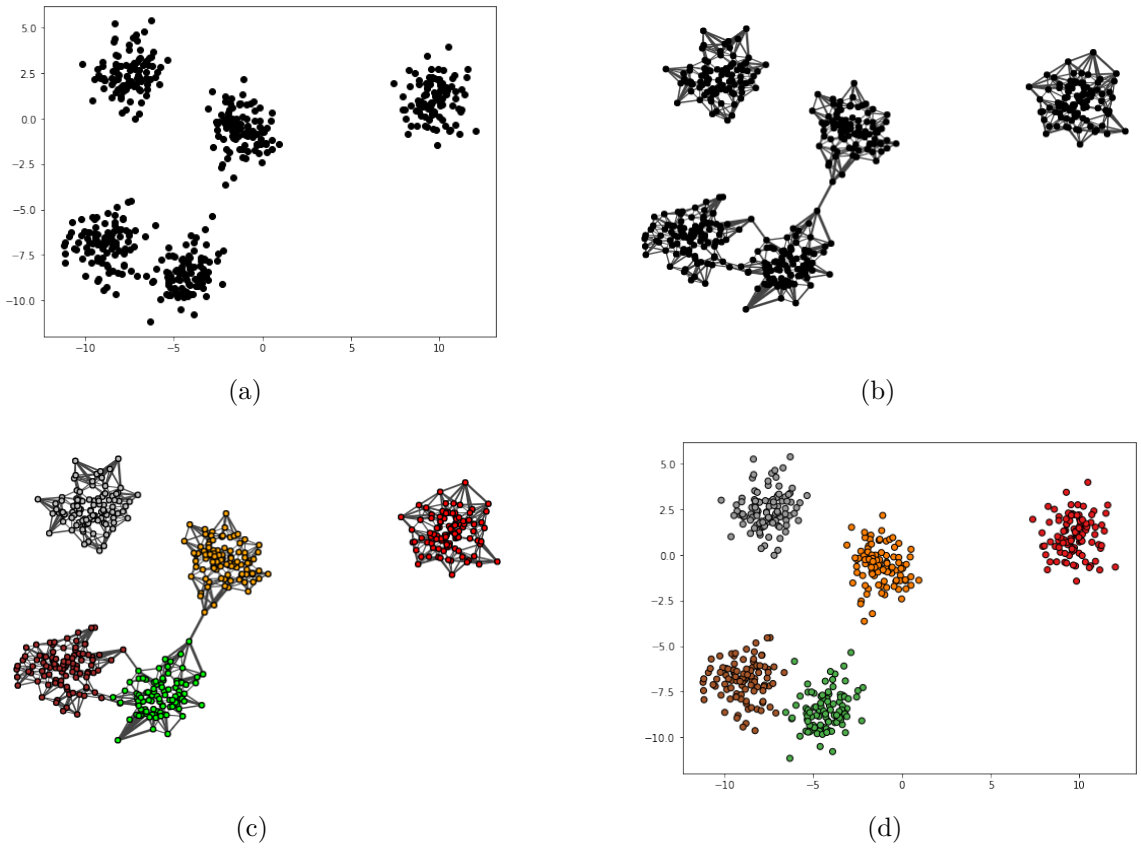


Figura 1 – Processo de agrupamento baseado em redes utilizando detecção de comunidades (Do autor).

a métrica de avaliação selecionada para fins de comparação e o conjunto de dados utilizados.

1.1.2 Específicos

- Pesquisar estado da arte sobre análises comparativas de algoritmos de agrupamento e detecção de comunidade;
- Realizar experimentos executando ambos os tipos de algoritmos em dados gerados artificialmente;
- Calcular as métricas de avaliação para todos os algoritmos utilizados;
- Verificar a hipótese estabelecida no objetivo geral do trabalho;
- Avaliar o impacto de diferentes algoritmos de construção de redes no desempenho dos algoritmos de detecção de comunidades.

1.2 Justificativa

Este trabalho apresenta uma visão geral do processo de agrupamento, realizando uma análise entre as abordagens mais comuns atualmente. O intuito é documentar qual abordagem tem melhor desempenho levando em consideração questões que até então não foram exploradas na literatura. Apesar de haver extensa pesquisa e documentação dos algoritmos aqui estudados, ainda há uma lacuna na literatura para compará-los. Na revisão bibliográfica foram encontrados trabalhos diretamente relacionados com o tema aqui proposto, sendo os principais deles: "Uma Análise Comparativa entre Técnicas de Detecção de Comunidades com Aplicação para o Problema de Agrupamento de Objetos Invariantes" [13] e "Um estudo computacional comparativo entre algoritmos de agrupamento e de detecção de comunidades" [41].

No primeiro caso, o trabalho tem um grande foco no desempenho de algoritmos de detecção de comunidades, não possuindo ampla experimentação com algoritmos de agrupamento como forma de comparação. Além disso, também é desconsiderado o impacto dos algoritmos de construção de redes no desempenho final. Por outro lado, o segundo trabalho realiza uma análise entre os dois grupos de algoritmos, porém com foco em outras medidas de desempenho, desconsiderando também o impacto dos algoritmos de construção de redes escolhidos.

Dito isso, ao alcançar os objetivos citados na seção anterior, este trabalho tem como justificativa preencher a lacuna na bibliografia citada acima.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma: No capítulo 1 é apresentada a proposta de pesquisa, os objetivos e a justificativa para realizá-lo. No capítulo 2 é realizada uma revisão bibliográfica, fazendo o levantamento de conceitos importantes de Aprendizagem de Máquina e os algoritmos utilizados nos experimentos. No capítulo 3 é apresentada a metodologia para realização dos experimentos, bem como as ferramentas utilizadas e bases de dados utilizados. O capítulo 4 apresenta a análise dos resultados obtidos e a análise comparativa entre os algoritmos. Por fim, no capítulo 5 é descrito as conclusões do trabalho, bem como os próximos passos que podem ser explorados em trabalhos futuros.

2 Fundamentação teórica

Este capítulo apresenta os principais conceitos de aprendizado de máquina com foco nos algoritmos de agrupamento e detecção de comunidades, objeto de estudo deste trabalho.

2.1 Aprendizagem de máquina

Aprendizado de Máquina (AM) é uma subárea da Inteligência Artificial (IA) que busca desenvolver modelos que possam "aprender" através da experiência. AM estuda métodos computacionais para adquirir novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente [27]. O aprendizado se dá por meio de algoritmos que, baseados em estatística, identificam regras e padrões em grandes bases de dados. O estudo de técnicas de aprendizado baseado em computador também pode fornecer um melhor entendimento de nosso próprio processo de raciocínio [28].

De forma arbitrária, uma das grandes críticas à IA é que as máquinas só podem ser consideradas inteligentes uma vez que forem capazes de aprender novos conceitos e se adaptarem a novas situações de forma independente, em vez de simplesmente executar o que lhes foi programado. Ada Lovelace, uma das primeiras filósofas em computação, escreveu "A Máquina Analítica não tem qualquer pretensão de originar nada. Ela pode fazer qualquer coisa desde que nós saibamos como mandá-la executar". Esse pensamento pode ser interpretado como indicação de que os computadores não são capazes de aprender, entretanto, é possível criar um sistema computacional capaz de interpretar as informações recebidas e aprender de modo a melhorar o seu desempenho por meio da observação e da experiência.

Existem várias abordagens de aprendizado que podem ser utilizadas por um sistema computacional, como aprendizado por hábito, por instrução, por dedução, por analogia e por indução. As técnicas de AM utilizam a abordagem de aprendizado indutivo, que consiste em obter conclusões genéricas a partir de um conjunto particular de exemplos ou casos particulares previamente observados. Na indução, um conceito é aprendido efetuando-se inferência indutiva sobre os exemplos apresentados. Portanto, as hipóteses geradas através da inferência indutiva podem ou não preservar a verdade [35]. Na Figura 2 é mostrada a hierarquia do aprendizado indutivo.

O aprendizado indutivo pode ser dividido, de forma mais geral, em dois tipos principais que serão descritos a seguir: supervisionado e não-supervisionado.

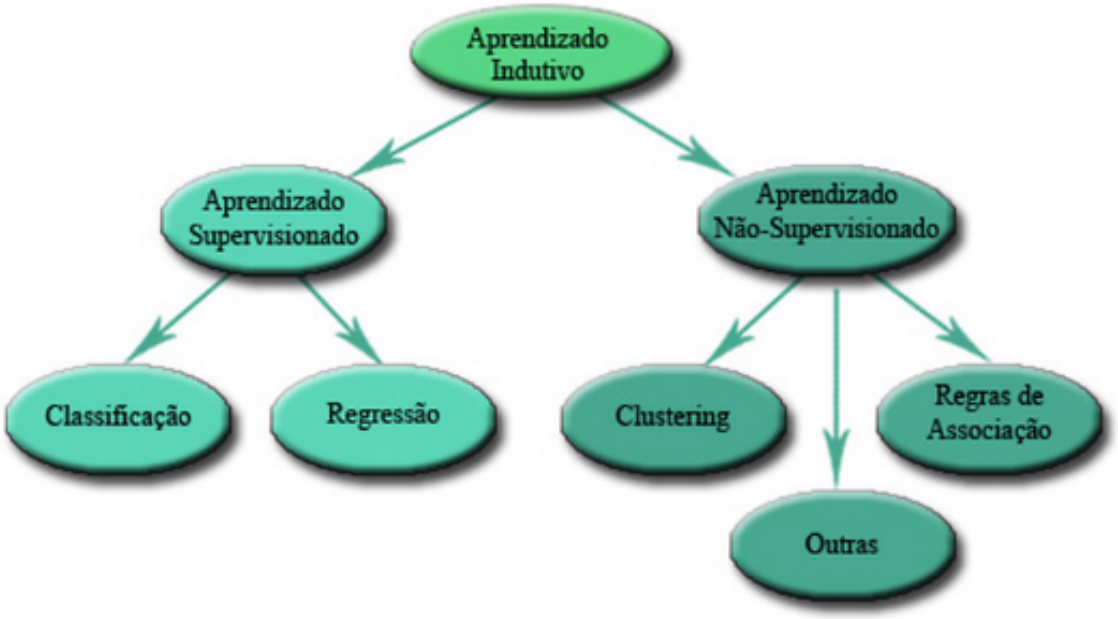


Figura 2 – Hierarquia do aprendizado indutivo [35].

2.1.1 Aprendizagem supervisionada

Aprendizagem de máquina supervisionada consiste na criação de um classificador capaz de aprender a partir de um conjunto de treinamento e generalizar para as novas instâncias que aparecem [22]. Neste tipo de AM, tem-se a figura de um "professor externo", que apresenta o conhecimento do ambiente por conjuntos de exemplos na forma: entrada e saída desejada [17]. Dessa forma ele possuirá um escopo limitado e pré-definido de resultados que serão utilizados como parâmetro e, principalmente, como referência durante a classificação.

Classificação e regressão são dois tipos principais de problemas de aprendizado de máquina supervisionados [16]. A diferenciação entre tarefas de classificação e regressão pode ser feita ao se perguntar se existe algum tipo de continuidade na saída. Se houver continuidade entre os resultados possíveis, se trata de um problema de regressão, caso contrário, podemos definir um problema de classificação, embora essa definição seja mais genérica.

2.1.2 Aprendizagem não-supervisionada

Os casos em que permanece a necessidade de análise dos dados mesmo sem eles estarem rotulados são considerados de aprendizado não supervisionado, onde se aprende sem um professor. Desconhecer a "resposta certa" é o melhor caminho para o aprendizado

não supervisionado [9].

A classificação através do método não supervisionado não apresenta exemplos da saída desejada, nesse caso o algoritmo de AM aprende a representar (ou agrupar) as entradas submetidas segundo uma estrutura natural presente nos dados e deve possuir recursos para superar essa falta de respostas através da apresentação dos resultados classificados de acordo com categorias formuladas por ele próprio.

O tipo de aprendizado abordado neste trabalho é o não-supervisionado, com ênfase em algoritmos de agrupamento e detecção de comunidades, que serão abordados a seguir.

2.2 Algoritmos de agrupamento

Agrupamentos são classes ou grupos de objetos que compartilham características em comum e desempenham um papel importante em como as pessoas analisam e descrevem o mundo.

No contexto de computação, para a extração de conhecimento de uma base de dados é fundamental separar estes dados em forma de grupos que possuam algum significado relevante para a análise. A criação destes grupos é necessária para que se realize uma melhor investigação do grande volume de dados.

Embora a ideia do que se constitui um grupo seja intuitiva, não existe definição formal única para esse conceito, ou seja, há diversas noções de um grupo que se provam úteis na prática em diferentes situações e contextos. Isso é resultado da grande diversidade de visões/objetivos de diferentes áreas que utilizam/desenvolvem técnicas de agrupamento. Algumas das definições presentes na literatura são:

> Um grupo é um conjunto de entidades semelhantes e entidades pertencentes a grupos diferentes não semelhantes [21].

> Grupos podem ser descritos como regiões conectadas de um espaço multidimensional contendo uma alta densidade relativa de pontos, separados de outras regiões por uma região contendo uma baixa densidade relativa de pontos [12].

Agrupamento também é um processo subjetivo, deste modo, é necessário atenção extra ao se realizar uma análise de grupo nos dados. A subjetividade está presente em diversos aspectos, entre eles nas hipóteses estabelecidas sobre os dados, a definição da medida de proximidade, a determinação do número de grupos, a seleção do algoritmo de agrupamento e a determinação dos índices de validação [45].

O processo de agrupamento compreende diversas etapas que vão desde a preparação dos objetos, até a interpretação dos grupos obtidos. A Figura 3 resume as etapas do

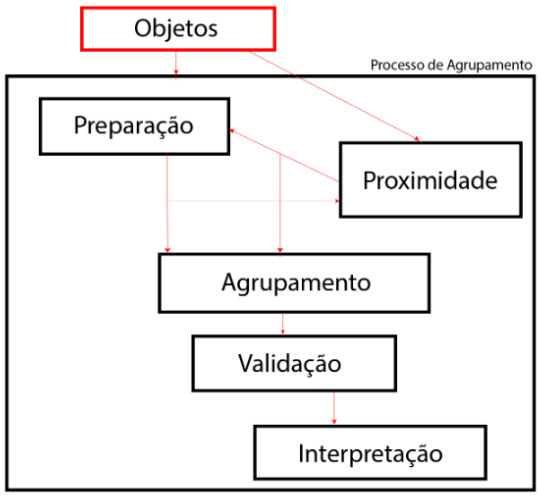


Figura 3 – Etapas do Processo de Agrupamento [26].

processo de agrupamento com as informações utilizadas e geradas em cada etapa. As etapas e a figura apresentada são baseadas nas informações apresentadas por [21] e [3], cada uma dessas etapas são descritas a seguir.

Preparação: A preparação dos dados para agrupamento envolve aspectos relacionados ao seu pré-processamento, como normalizações, conversão de tipos, redução do número de atributos por meio de seleção ou extração de características [21], e à forma de representação para sua utilização em um algoritmo de agrupamento.

Proximidade: Consiste na definição de uma medida de proximidade apropriada ao domínio da aplicação. Essa medida pode ser uma medida de similaridade ou de dissimilaridade entre dois objetos. A escolha da medida de proximidade a ser empregada com um algoritmo de agrupamento deve considerar os tipos e escalas dos atributos que definem os objetos e também as propriedades dos dados que o pesquisador deseja focalizar. Uma das medidas de proximidade mais comumente utilizada é a Distância Euclidiana, formalizada pela seguinte equação:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Agrupamento: Esta etapa consiste da aplicação de um algoritmo de agrupamento. essas técnicas serão vistas nas próximas seções deste trabalho.

Validação: Esta etapa se refere à avaliação do resultado de um agrupamento e deve, de forma objetiva, determinar se a solução é representativa para o conjunto de dados analisado. Uma estrutura de agrupamento é válida se não ocorreu por acaso ou se

é "rara" em algum sentido, já que qualquer algoritmo de agrupamento encontrará grupos, independentemente de existir ou não similaridade nos dados [21].

Interpretação: Refere-se ao processo de examinar cada grupo com relação a seus objetos para rotulá-los, descrevendo a natureza do grupo. A interpretação de grupos é mais que apenas uma descrição. Além de ser uma forma de avaliação dos grupos encontrados e da hipótese inicial, de um modo confirmatório, os grupos podem permitir avaliações subjetivas que tenham um significado prático.

Existem diversos algoritmos de agrupamentos descritos na literatura, porém, não existe um algoritmo de agrupamento universal capaz de revelar todos os tipos de estruturas que podem estar presentes em um conjunto de dados. Além da dificuldade da escolha do melhor algoritmo para determinado contexto ou aplicação, muitos algoritmos apresentam limitações. Os problemas comuns a vários algoritmos de agrupamento descritos por [21] são:

- Adequação a domínios e/ou conjuntos de dados restritos.
- Restrição dos formatos da estrutura que pode ser encontrada.
- Necessidade de conhecimento prévio do número de grupos presentes nos dados ou o difícil ajuste de parâmetros.
- Instabilidade dos resultados obtidos. Várias execuções de um algoritmo produzem agrupamentos diferentes, podendo associar um mesmo objeto a grupos diferentes.

2.3 Algoritmos de agrupamento tradicionais

2.3.1 Algoritmo K-Means

O algoritmo K-Means é uma das técnicas de agrupamento mais conhecidas e possui o maior número de variações devido à sua simplicidade e facilidade de implementar em linguagens computacionais.

Este algoritmo requer que o número de grupos seja especificado inicialmente. Ele se adapta bem a um grande número de exemplos e de grupos e tem sido usado em uma grande variedade de aplicação em diferentes áreas.

O algoritmo K-Means implementa uma técnica de agrupamento baseada em protótipos (centros). O algoritmo inicia escolhendo k centroides iniciais, em que k é o número de grupos definido pelo usuário. Cada padrão é então atribuído ao centroide mais próximo, e cada coleção de padrões atribuída ao centroide forma um grupo. O

centroide de cada grupo é então atualizado baseado nos padrões atribuídos ao grupo. O processo de atribuição dos padrões e a atualização dos centroides se repete até que os centroides permaneçam inalterados. Este processo pode ser observado no algoritmo descrito na Figura 4.

Algoritmo 2 Algoritmo K-Means.

1: especifique k.
2: selecione os k objetos que serão os centroides dos agrupamentos.
3: **enquanto**
4: **Para** *todos os objetos Faça.*
5: *Calcule a distância entre o objeto e os centroides.*
6: *Adiciona o objeto ao grupo que possuir a menor distância.*
7: *Recalcule os centroides dos grupos.*
8: **Fim do Para**
9: **até que** *o número de interações* $= i$ *ou Não ocorra mudança dos centroides.*

Figura 4 – Algoritmo de Agrupamento K-Means [26].

A Figura 5 mostra um exemplo prático dos passos da aplicação do algoritmo K-Means. No primeiro passo, é definido um número k de centros arbitrários e então cada objeto é incluído em grupo do centro mais próximo. Posteriormente, o centro dos grupos gerados são atualizados e o processo se repete até que os centros permaneçam inalterados.

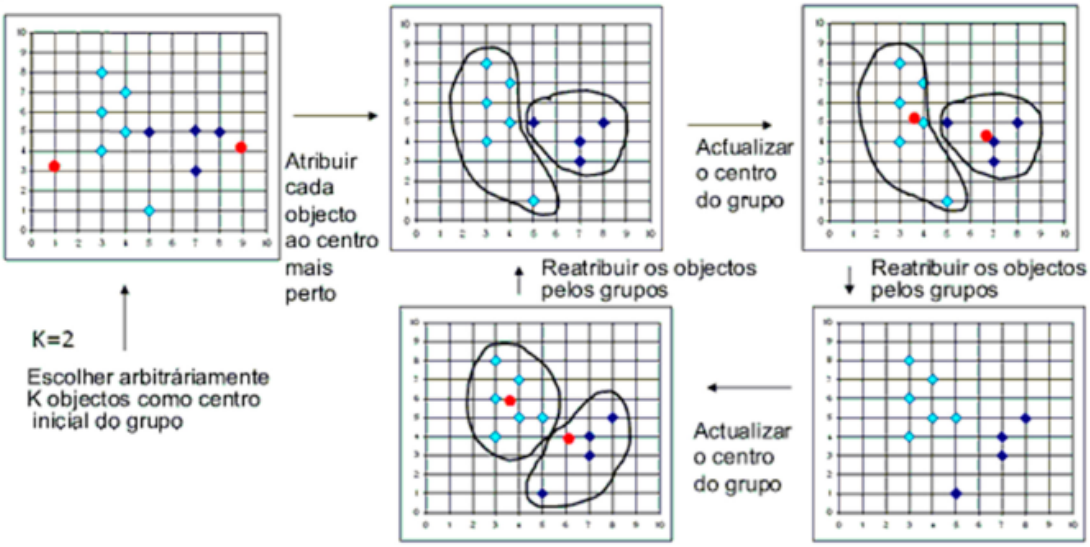


Figura 5 – Passos de aplicação do algoritmo K-Means [26].

2.3.2 DBSCAN

DBSCAN, abreviação do termo "*Density Based Spatial Clustering of Application with Noise*" (Agrupamento Espacial Baseada em Densidade de Aplicações com Ruído) é um método de agrupamento não paramétrico baseado em densidade (número de pontos dentro de um raio específico) (EPS), proposto no ano de 1996 por [11].

Agrupamentos baseados em densidade localizam regiões de alta densidade que estejam separadas entre si por regiões de baixa densidade. Os dois principais parâmetros de entrada que o DBSCAN necessita são:

Raio de vizinhança de um ponto (EPS): determina o raio de vizinhança para cada ponto da base de dados. Dado o parâmetro EPS, o algoritmo DBSCAN verifica a quantidade de pontos contidos no raio (EPS) para cada ponto da base de dados, e se essa quantidade exceder certo número, um grupo é formado;

Número mínimo de pontos (MinPts): parâmetro que especifica o número mínimo de pontos, no dado raio (EPS), que um ponto precisa possuir para ser considerado um ponto central e consequentemente, de acordo com as definições de grupo baseado em densidade, inicia a formação de um grupo.

Com os parâmetros EPS e MinPts definidos o algoritmo basicamente realiza a separação do conjunto de observações em três classes:

Pontos Centrais: Estes pontos estão no interior de um grupo baseado em densidade. Um ponto é central se o número de pontos dentro de uma determinada vizinhança em torno do ponto conforme determinado pela função de distância e um parâmetro de distância especificada pelo usuário, EPS, exceder um determinado limite, MinPts. Na Figura 6, o ponto A é um ponto central, para o raio indicado EPS se MinPts $<= 7$.

Pontos Limites: Um ponto de limite não é um ponto central, mas fica dentro da vizinhança de um ponto central. Na Figura 6, o ponto B é um ponto de limite. Um ponto de limite pode cair dentro das vizinhanças de diversos pontos centrais.

Pontos de Ruídos: Um ponto de ruído é qualquer ponto que não seja nem um ponto central nem um ponto limite. Na Figura 6, o ponto C é um ponto de ruído.

Dadas as definições de pontos de centro, de limite e de ruído o algoritmo DBSCAN pode ser descrito informalmente da seguinte maneira: Quaisquer dois pontos do centro que estejam suficientemente próximos, dentro de uma distância EPS entre si, são colocados no mesmo grupo. Da mesma forma, qualquer ponto de limite que esteja

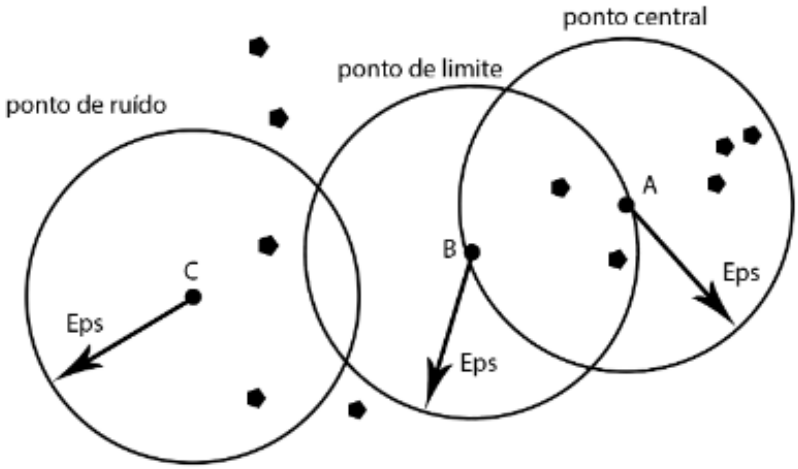


Figura 6 – Pontos de centro, de limite e de ruído [26].

suficientemente próximo de um ponto do centro é colocado no mesmo grupo do ponto de centro. Pontos de ruído são descartados. O processo descrito para o DBSCAN pode ser observado no algoritmo presente na Figura 7.

Algoritmo 3 Algoritmo DBSCAN.

```
1:  Rotular todos os pontos como de centro, de limite ou ruído.
2:  Eliminar os pontos de ruído.
3:  Colocar uma aresta entre todos os pontos de centro que...
3:  estejam dentro da Eps uns dos outros.
4:  Tornar cada grupo de pontos de centro conectados um grupo separado.
5:  Atribuir cada ponto de limite a um dos grupos dos seus pontos de centro associados.
```

Figura 7 – Algoritmo de Agrupamento DBSCAN [26].

Como os valores dos parâmetros EPS e MinPts influenciam diretamente no resultado do algoritmo, é preciso determinar uma forma de calculá-los para realização dos experimentos neste trabalho. Não existe na literatura uma forma definitiva de defini-los, pois, dependem da natureza e contexto do problema, porém, é possível encontrar algumas indicações.

Para o parâmetro **MinPts** é indicado que o valor seja maior ou igual ao número de dimensões das amostras da base de dados [29]. Na seção de experimentos, será detalhada a base de dados utilizadas, porém, tendo em vista que serão dados bidimensionais, é possível adotar um valor fixo MinPts = 2.

Em relação ao parâmetro **EPS**, pode ser escolhido usando um gráfico de k-distância, traçando o gráfico da distância para o vizinho mais próximo k = minPts-1 ordenado do

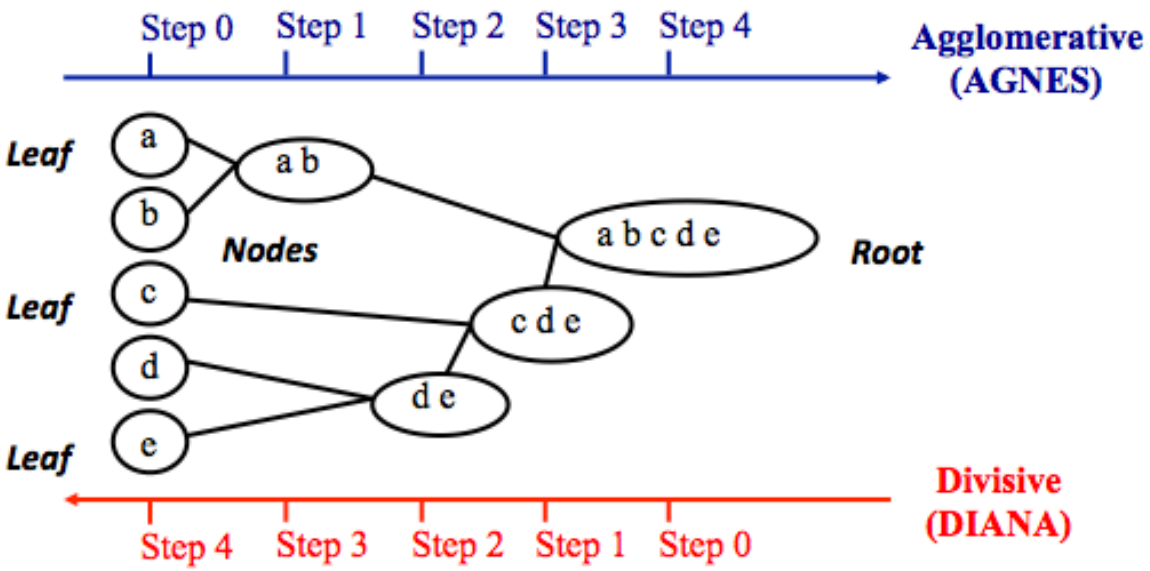


Figura 8 – Agglomerative Clustering [8].

maior para o menor valor. Bons valores de EPS estão onde aparece um "cotovelo" neste gráfico [29]. Se o EPS for escolhido muito pequeno, uma grande parte dos dados não será agrupada; enquanto que para um valor muito alto de EPS os grupos se fundirão e a maioria dos objetos estará no mesmo grupo. Como este método depende de uma análise visual do gráfico de k-distância, o algoritmo de classificação DBSCAN será executado com diferentes valores de EPS para avaliar seus respectivos desempenhos e impactos em diferentes bases de dados.

2.3.3 Agglomerative Clustering

Agglomerative Clustering é uma classe de algoritmos que faz parte de um grupo de algoritmos de agrupamento ditos "hierárquicos". O agrupamento hierárquico, como o nome sugere, é um algoritmo que constrói a hierarquia de grupos. Esse algoritmo começa com todos os pontos de dados atribuídos a um grupo próprio. Em seguida, dois grupos mais próximos são mesclados no mesmo grupo. No final, esse algoritmo termina quando há apenas um único grupo.

O agrupamento aglomerativo funciona de maneira "bottom-up". Ou seja, cada objeto é inicialmente considerado um grupo de um único elemento (folha). Em cada etapa do algoritmo, os dois grupos mais semelhantes são combinados em um novo grupo maior (nós). Este procedimento é iterado até que todos os pontos sejam membros de apenas um único grande grupo (raiz). Este processo está exemplificado na Figura 8.

Neste trabalho, o algoritmo aglomerativo que será abordado é o Ward, que minimiza a soma das diferenças quadradas em todos os grupos. É uma abordagem de minimização

de variância e, neste sentido, é semelhante à função objetivo KMeans, mas abordada com uma abordagem hierárquica aglomerativa [31]. Este algoritmo também pode escalar para um grande número de amostras quando é usado em conjunto com uma matriz de conectividade, mas é computacionalmente caro quando nenhuma restrição de conectividade é adicionada entre as amostras, pois considera em cada etapa todas as combinações possíveis.

2.4 Algoritmos de agrupamento baseados em rede

2.4.1 Definições Básicas

Qualquer situação onde temos objetos e relações entre eles, é possível representar esse conjunto de dados por meio de uma rede. Uma rede é uma estrutura relacional na qual os vértices representam os objetos e as arestas representam as relações existentes entre os objetos.

Uma rede $G(V, E)$ representa um conjunto finito $V = v_1, v_2, ..., v_n$, cujos n elementos são denominados vértices, juntamente com um conjunto $E = e_1, e_2, ..., e_m$ é o conjunto de arestas, tal que uma aresta que liga os vértices v_i e $v_j \in V$ é denotada por $e_{i,j}$. Uma rede não direcionada é dita simples se cada par de vértices está ligado, no máximo, com uma aresta, ou seja, não existem arestas paralelas.

Uma rede ponderada $G(V, E, W)$ representa um conjunto de vértices V, um conjunto de arestas E e uma matriz de peso W, sendo w uma função que atribui para cada aresta $e_{i,j} \in E$ um peso $w(e_{i,j})$.

Dois vértices são ditos adjacentes ou vizinhos se estão conectados por uma aresta. A matriz de adjacência A de uma rede é uma matriz quadrada de ordem n, sendo $n = |V|$, cuja entrada $a + i, j$ representa uma aresta entre o vértice v_i e o vértice v_j.

Além da representação de uma rede por meio de sua matriz de adjacência, é possível representar a rede através de uma lista de adjacência, que consiste em um vetor adj de n listas, uma para cada vértice de V. A Figura 9 ilustra uma rede simples e sua representação usando matriz de adjacência e lista.

2.4.2 Construção de Redes

Diversas áreas de aprendizado de máquina (AM) utilizam redes para modelar relações locais entre dados. Dado este cenário, uma das principais questões relacionadas com o aprendizado baseado em redes trata de qual técnica de construção de redes utilizar para cada aplicação e como definir seus parâmetros, sendo muitas vezes mais importante a escolha do método de construção de redes que o algoritmo de aprendizado em si [46].

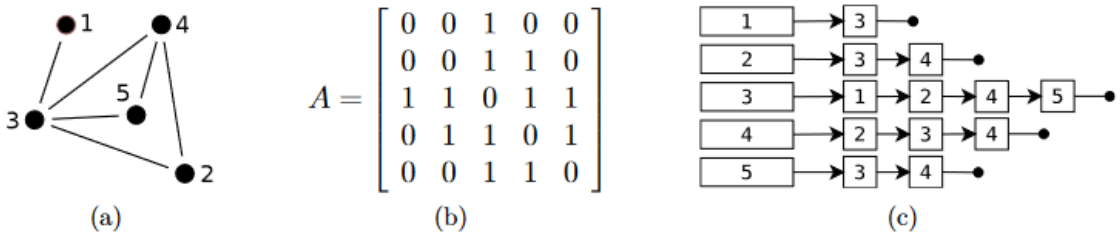


Figura 9 – Modelagem da rede (a) por meio de uma Matriz de Adjacência (b) e Lista de Adjacência (c) [44].

Dependendo do paradigma de AM, alguns modelos são mais adequados que outros. No aprendizado supervisionado as redes têm como restrição não estabelecer conexões entre elementos da mesma classe, enquanto que para o aprendizado não-supervisionado deseja-se que o número de conexões intra-grupos seja maior que o número de conexões entre-grupos.

Veremos nessa seção que os algoritmos de detecção de comunidades se baseiam em dados estruturados no formato de redes, por isso, para realização da análise comparativa entre algoritmos de agrupamento e de detecção de comunidade, se faz necessário utilizar técnicas de construção de redes a partir de uma base de dados.

A seguir são apresentados os principais métodos para construções de redes a partir do conjunto X.

2.4.2.1 KNN

O algoritmo KNN, também conhecido como K-vizinhos mais próximos, foi proposto para o aprendizado supervisionado. Dado um exemplo $x_i = (x_{i,j}, \ldots, x_{i,m})$ que deve ser rotulado, o algoritmo KNN obtêm-se os k vizinhos mais similares a x_j.

Para construção de redes, inicialmente, o algoritmo KNN calcula a distância para todos os pares de objetos x_i e x_j. Várias medidas de distâncias podem ser utilizadas, por exemplo, a distância euclidiana ou o cosseno. Em seguida, cada objeto x_i é considerado um vértice v_i na rede resultante. Por fim, o KNN gera uma conexão entre os vértices v_i e v_j, se a distância entre v_i e v_j for menor ou igual a k-ésima distância a partir de v_i e os demais objetos do conjunto.

A Figura 10 demonstra o processo de geração das redes KNN usando diferentes valores de k. É possível notar que um valor baixo de k gera uma rede desconexa e, conforme o valor de k é incrementado, a densidade de arestas na rede aumenta. Além disso, a rede KNN força um vértice a se conectar com seus k vizinhos mais próximos, mesmo que estejam distantes, o que significa que a vizinhança desse vértice pode conter exemplos dissimilares.

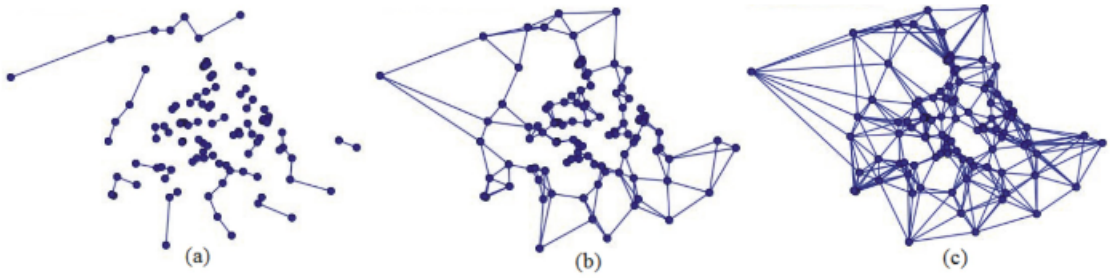


Figura 10 – Exemplo de redes KNN para um conjunto de dados com 100 elementos e distribuição gaussiana (a) k=1, (b) k=3 e (c) k=7. [24].

2.4.2.2 Mutual KNN

O algoritmo Mutual KNN, também conhecido como Mk-NN, se trata de uma variação do algoritmo KNN. Nesse caso, um vértice estabelece uma ligação somente se a regra de vizinhança se cumprir mutuamente, ou seja, existirá uma aresta entre v_i e v_j se, e somente se, $v_i \in KNN(v_j)$ e $v_j \in KNN(v_i)$, tal que $KNN(v_i)$ retorna os k-vizinhos mais próximos de v_i. Por conta disso, as redes Mk-NN são consideradas mais restritivas, uma vez que possuem regras extras para formação das redes, sendo utilizadas tradicionalmente no aprendizado não supervisionado [15]; [6].

A Figura 11 é um exemplo de uma rede gerada pelo algoritmo Mk-NN a partir de uma base de dados gerada de forma uniformemente distribuída em duas regiões poligonais [6].

Figura 11 – Geração de rede com Mk-NN (a) Dados e (b) Rede [6].

De acordo com Maier [25], observaram que as redes Mk-NN favorecem a identificação de grupos, principalmente em casos que os grupos possuem diferentes densidades e pesos.

Já as redes KNN são mais apropriadas para identificar vários grupos simultaneamente, por terem melhores propriedades de conectividade.

2.4.2.3 E-Vizinhança

No algoritmo E-Vizinhança, também referido nesse trabalho como Epsilon, cria-se uma aresta entre os objetos x_i e x_j se $|(x_i, x_j)| \leq \epsilon$, sendo $|(x_i, x_j)|$ uma função de distância entre dois pontos e ϵ um limiar pré-definido. Portanto, dado um ponto, cria-se uma circunferência de raio ϵ ao redor dele e são adicionadas arestas conectando esse ponto com todos os outros dentro da circunferência. Percebe-se que, ao utilizar essa técnica, se ϵ for um valor muito baixo, deve-se fazer um pós-processamento para que não haja vértices isolados na rede, já se ϵ for um valor muito alto a rede gerada pode ser muito densa. Este comportamento está exemplificado na Figura 12.

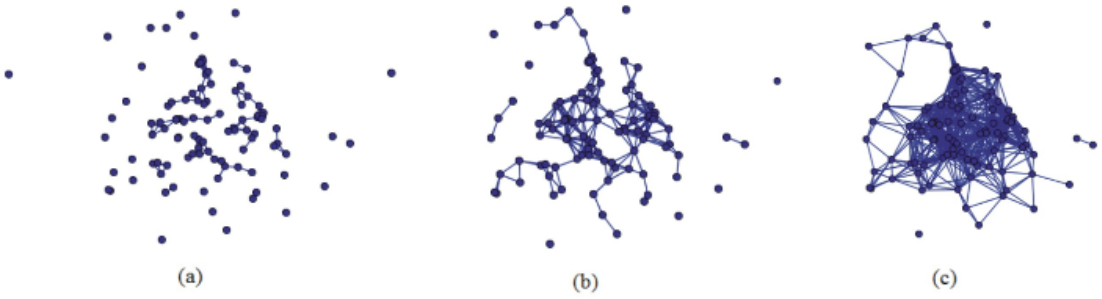


Figura 12 – Exemplo de redes E-Vizinhança para um conjunto de dados com 100 elementos e distribuição gaussiana (a) E=0.3, (b) E=0.5 e (c) E=0.9. [24].

Devido a essa desvantagem, as redes E-Vizinhança têm sido pouco utilizadas. Segundo Belkin e Niyogi, 2003 [5] esse algoritmo resulta facilmente em componentes desconexos, não sendo adequados para diversas situações.

2.4.3 Detecção de comunidades

Em redes complexas, comunidades representam a organização de vértices em grupos, que podem ser identificados pela existência de muitas arestas ligando vértices de um mesmo grupo, que provavelmente possuem propriedades comuns ou desempenham função similares na rede, e poucas arestas ligando vértices de grupos diferentes. O processo de encontrar as estruturas naturais de comunidades em uma rede é chamado de Detecção de Comunidades [14]; [2].

Um exemplo real de aplicação de Detecção de Comunidades são os sistemas de compras online, onde agrupar os clientes por semelhança de interesses pode proporcionar a criação de sistemas de recomendações mais eficientes [14]; [2].

Os métodos de Detecção de Comunidades se diferenciam dos métodos de agrupamentos tradicionais por levar em consideração não apenas um critério de similaridade entre os dados, mas também a sua topologia, estrutura ou dinâmica. Tal característica é importante pois permite capturar vários padrões pela análise de funcionalidades e processos operando sobre a rede.

Nos métodos de agrupamento tradicionais, o algoritmo recebe um conjunto com os atributos dos dados e calcula a similaridade dos dados baseado em algumas medidas de distância presentes na literatura. Em redes, esse conjunto de dados é utilizado para a formação da rede e então são aplicados os métodos de Detecção de Comunidades sobre a rede formada.

2.4.3.1 Fastgreedy

O algoritmo Fastgreedy, [30]; [7], que faz parte da classe de algoritmos hierárquicos aglomerativos, sendo amplamente utilizado, é baseado na ideia de modularidade.

Considerando que não é esperado uma estrutura de comunidades em uma rede aleatória, o cálculo da modularidade é feito baseado na comparação entre a densidade de arestas dentro das comunidades e a densidade esperada em uma rede aleatória. Assume-se que valores altos de modularidade representam uma boa divisão da rede [14].

A Modularidade calcula um valor para uma determinada divisão da rede, e quanto maior esse valor, mais bem divididos são os grupos na rede. Se altos valores de modularidade correspondem a uma melhor divisão da rede em comunidades, uma possível opção de algoritmo é gerar todas as possíveis divisões da rede, calcular a modularidade para cada uma e encontrar a divisão que resulta em um maior valor para a modularidade. Essa estratégia considera uma rede desconexa com n vértices, na qual, inicialmente, cada vértice representa uma comunidade distinta.

Entretanto, como destacado por Clauset et al. (2004) [7], considerando que as possibilidades de divisão da rede crescem exponencialmente ao número de vértices, encontrar todas as divisões possíveis da rede não seria uma tarefa simples, portanto, deve-se utilizar uma heurística para minimizar as divisões geradas. O algoritmo Fastgreedy utiliza a heurística gulosa que é sempre tentar descartar as possibilidades que não geram o melhor resultado.

O algoritmo inicia com cada vértice da rede sendo uma comunidade (singletons) e a cada iteração pares de comunidades são unidas. A escolha dos pares é feita de forma a maximizar o valor da modularidade. O processo de união é repetido até que todos os vértices façam parte de uma única comunidade. As uniões obtidas a cada iteração são armazenadas em uma estrutura hierárquica denominada dendrograma. Após todas as execuções, o dendrograma gerado é analisado e se escolhe a divisão que produz o melhor

valor para a modularidade. A Figura 13 é uma ilustração de um dendrograma com o corte que representa o valor máximo da modularidade.

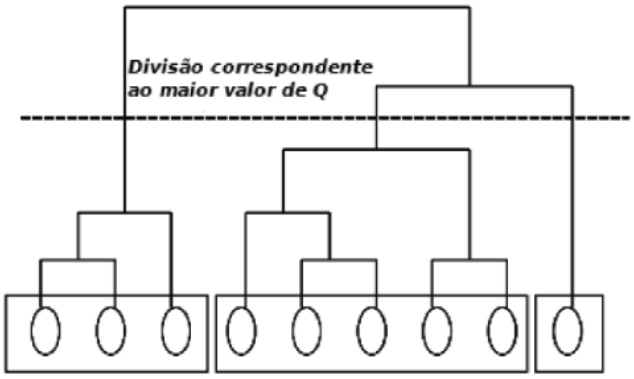


Figura 13 – Ilustração de um dendograma com o corte que representa o valor máximo da modularidade [34].

A Figura 14 apresenta o algoritmo Fastgreedy, considerando Q valores da modularidade e $\Delta Q_{i,j}$ a variação da modularidade obtida ao juntar duas comunidades i e j.

Entrada: rede G
1 *dendograma* $D \leftarrow \emptyset$;
2 calcular o valor inicial de Q para a rede;
3 calcular os valores de ΔQ para cada possível união;
4 preencher o *max-heap* H com os valores de ΔQ;
5 **repita**
6 | unir o par de comunidades i e j com maior $\Delta Q_{i,j}$ em H;
7 | atualizar D, ΔQ, *max-heap* e incrementar o valor Q com $\Delta Q_{i,j}$;
8 **até** *até obter apenas* 1 *comunidade*;
9 **retorna** D

Figura 14 – Algoritmo Fastgreedy [44].

2.4.3.2 Infomap

O algoritmo Infomap é parecido com o algoritmo Fastgreedy, e também é considerado da classe hierárquico aglomerativo, entretanto, ao invés de maximizar a Modularidade, ele tenta minimizar a chamada *Map Equation*, baseada na entropia [36].

A *Map Equation* explora a dualidade entre encontrar vértices densamente conectados (comunidades) e minimizar o comprimento de uma chamada "caminhada aleatória". Uma caminhada aleatória se caracteriza por ser um processo estocástico utilizado como

um mecanismo de transporte e pesquisa em redes. A caminhada aleatória é performada conforme calcula-se uma probabilidade para que se siga do vértice atual em direção a outro vértice ou se mantenha no vértice atual, sendo que tal probabilidade é proporcional ao peso da aresta em questão. Dessa forma, uma caminhada aleatória tende a ficar "presa" com maior frequência em regiões densamente conectadas, nas quais vértices podem compartilhar propriedades e desempenhar papeis semelhantes, e as caminhadas entre as regiões esparsamente conectadas são relativamente raras. Essas regiões densamente conectadas encontradas pela caminhada aleatória são consideradas comunidades.

Entretanto, a caminhada aleatória por si só pode não gerar o número de comunidades ideal e é nesse ponto que age a *Map Equation*, que pode comprimir o comprimento de uma caminhada aleatória para que comunidades maiores sejam atingidas. Mais especificamente, cada vértice inicia como sendo uma única comunidades e, em cada iteração, em uma ordem sequencial aleatória, cada vértice é movido para a comunidade vizinha que resulta na minimização da *Map Equation*. Se nenhum movimento resultar em um ganho, o vértice permanece em sua comunidade original. Este procedimento é repetido, cada vez em uma nova ordem sequencial aleatória, até que nenhum movimento gere um ganho na *Map Equation*. Em um próximo nível, os vértices pertencentes a uma mesma comunidade são contraídos nos chamados super-vértices e a rede é compactada e reconstruída. Portanto, em cada nível, temos uma rede menor em relação ao nível anterior. Esta reconstrução da rede é repetida até que a *Map Equation* não possa ser reduzida ainda mais. Por conta dessa estratégia, esse algoritmo também é chamado de multinível.

A Figura 15 mostra uma rede ponderada com n=25 vértices. A espessura da conexão indica a probabilidade de que um caminho aleatório atravesse qualquer conexão específica.

2.4.3.3 Leading Eigenvector

O Leading Eigenvector faz parte de um grupo de algoritmos ditos espectrais. Os métodos espectrais são baseados na análise de autovalores, autovetores e propriedades algébricas por meio de representações matriciais de uma rede [44].

Dada uma matriz $M_{n,n}$, $\lambda \in \mathbb{R}$ é autovalor de M se existe um vetor $\gamma \neq 0$ tal que $M\gamma = \lambda\gamma$. Em decorrência, γ é um autovetor de λ.

Em teoria espectral, frequentemente, a representação da rede é dada por uma matriz Laplaciana, porém, o algoritmo Leading Eigenvector, proposto por [30], se baseia na matriz de modularidade B, definida por:

$$B_{v,u} = w_{v,u} - \frac{k_v k_u}{2m} \tag{2.1}$$

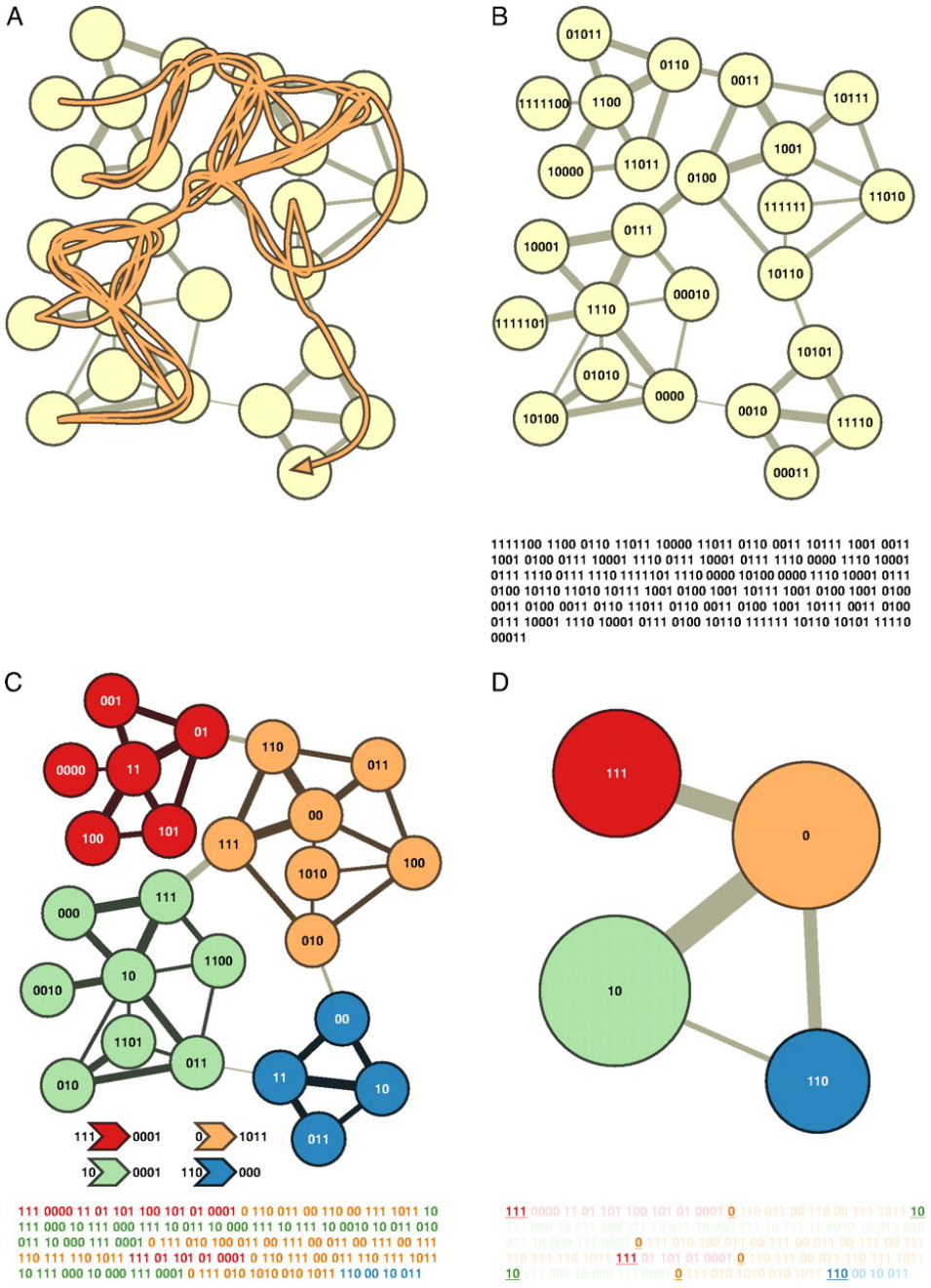


Figura 15 – Processo algoritmo Infomap [36].

onde $w_{v,u}$ corresponde ao peso da aresta entre os vértices v e u.

O método do Leading Eigenvector analisa o autovetor γ associado ao maior autovalor positivo de B. A separação de dois vértices é definida pelo sinal corresponde a cada vértice em γ. Vértices com sinais diferentes são incluídos em comunidades distintas. Caso todos os elementos de γ tenham o mesmo sinal, significa que a rede não possui estrutura de comunidades [44].

A Figura 16 apresenta um algoritmo genérico para um agrupamento espectral que pode ser extrapolado para o algoritmo Leading Eigenvector transformando a matriz Laplaciana L na matriz de modularidade B.

Entrada: rede G
1 *particionamento* $P \leftarrow \emptyset$;
2 crie a matriz de Laplace L a partir da rede G;
3 calcule os menores autovetores $\{\vec{\gamma}_1, ..., \vec{\gamma}_k\}$ de L associados aos menores autovalores;
4 interprete os autovetores $\{\vec{\gamma}_1, ..., \vec{\gamma}_k\}$ gerando o agrupamento P;
5 **retorna** P

Figura 16 – Algoritmo Genérico para Agrupamento Espectral [44].

2.5 Medidas de avaliação para agrupamento

Um dos desafios na aprendizagem não-supervisionada é avaliar quão bom é um determinado método, devido à ausência da categorização dos dados. Em agrupamento existem três tipos de medidas numéricas utilizadas para avaliar a qualidade dos grupos: índices internos, externos e relativos (Rendón et al., 2011). Índices do tipo externo comparam os grupos formados com uma estrutura conhecida previamente, ou seja, é necessário um conhecimento prévio sobre os dados para a comparação entre as estruturas previstas e geradas. Índices do tipo interno buscam avaliar informações intrínsecas aos grupos, sem fazer uso de quaisquer informações prévias. Já os índices relativos são usados para comparar diferentes soluções de agrupamento em relação a índices externos ou internos. Neste trabalho, será utilizado os índice externo Informação Mútua Normalizada (NMI - Normalized Mutual Informatial)

2.5.1 NMI

O NMI (*Normalized Mutual Information*) é uma normalização da pontuação de informação mútua entre as partições geradas por um algoritmo de agrupamento e as

classes reais (também chamada de partição de referência) dos objetos, onde o propósito é retornar um resultado entre 0 e 1, sendo 0 quando não há nenhuma informação mútua entre os conjuntos e 1 quando há uma perfeita correlação [42]. Portanto, essa medida é utilizada quando sabemos a divisão dos objetos previamente ou quando estamos avaliando dados artificiais e temos uma partição de referência.

A definição do cálculo para o NMI é feito da seguinte forma: Considere dois conjuntos, o conjunto predito (U) e o conjunto previsto (V), calculamos a entropia (H) para ambos os conjuntos:

$$H(U) = -\sum_{i=1}^{|U|} P(i)log(P(i)) \tag{2.2}$$

$$H(V) = -\sum_{j=1}^{|V|} P(j)log(P(j)) \tag{2.3}$$

Onde $P(i)$ é a probabilidade de um objeto aleatório de U ser atribuído à classe U_i, da mesma forma para $P(j)$. Após calcular as entropias, calcula-se o índice de informação mútua (MI) para os conjuntos U e V:

$$MI(U,V) = \sum_{i=1} |U| \sum j = 1|V|P(i,j)log(\frac{P(i,j)}{P(i)P'(j)}) \tag{2.4}$$

Onde $P(i,j) = |U \cap V|/N$ sendo o total de objetos. O MI também pode ser escrito da seguinte forma:

$$MI(U,V) = \sum_{i=1} |U| \sum_{j=1} |V| \frac{|Ui \cap Vj|}{N} log(\frac{N|Ui \cap Vj|}{|Ui||Vj|}) \tag{2.5}$$

O índice de informação mútua normalizado, Normalized Mutual Information (NMI), pode ser definido como:

$$NMI(U,V) = \frac{MI(U,V)}{media(H(U),H(V))} \tag{2.6}$$

3 Metodologia

Este capítulo descreve os métodos e técnicas, assim como os dados e ferramentas que foram utilizados para conduzir os experimentos. Todos os algoritmos utilizados e implementados utilizaram a linguagem de programação Python em conjunto com as bibliotecas do Scikit-learn (https://scikit-learn.org/stable/) e do igraph (igraph.org/python/). Resumidamente, os experimentos foram desenvolvidos por meio das seguintes etapas de elaboração do estudo:

1. Escolha dos algoritmos
2. Geração do conjunto de dados
3. Construção de redes
4. Realização dos experimentos
5. Avaliação dos resultados

3.1 Bibliotecas

Neste trabalho foram utilizados duas bibliotecas para realização dos experimentos: Scikit-learn e igraph. O Scikit-learn é um módulo Python que integra uma ampla gama de algoritmos de aprendizado de máquina de última geração para problemas supervisionados e não supervisionados. Este pacote se concentra em levar o aprendizado de máquina para não especialistas usando uma linguagem de alto nível de uso geral [33].

Além disso, foi utilizado também o igraph para implementação dos algoritmos referentes à construção de redes e detecção de comunidades. O igraph é uma coleção de bibliotecas para criação e manipulação e analise de redes. O software tem ênfase em eficiência, portabilidade e facilidade de utilização. Os algoritmos de detecção de comunidades utilizados nesse trabalho foram obtidos a partir dessa biblioteca.

3.2 Escolha dos algoritmos

Os algoritmos selecionados para realização dos experimentos foram detalhados nas seções anteriores, como parte da Revisão Bibliográfica. A seguir estão descritos os parâmetros fixados para implementação de cada um dos algoritmos:

K-Means: Implementação do scikit-learn.

- init: k-means++
- n_init: 10
- max_iter: 300

DBSCAN: Implementação do scikit-learn

- metric: euclidean
- min_samples: 2
- leaf_size: 30

Agglomerative Clustering: Implementação do scikit-learn

- affinity: euclidean
- linkage: ward
- compute_distances: falso

k-NN e Mk-NN: Implementação do scikit-learn: kneighbors_graph

- mode: distance
- metric: euclidean
- include_self: falso

E-Vizinhança: Implementação do igraph: radius_neighbors_graph

- mode: distance
- metric: euclidean
- include_self: falso

Fastgreedy, Infomap e Leading eigenvector: Implementação do igraph

- weights: Peso da rede ponderada

3.3 Geração do conjunto de dados

Para este trabalho foram utilizadas bases de dados artificiais obtidas através da função "*make blobs*" presente na biblioteca "*scikit-learn*". O *make blobs* cria conjuntos de dados multiclasse alocando a cada classe um ou mais grupos de pontos normalmente distribuídos.

A fim de gerar as bases de dados artificiais foi especificado, a principio, alguns parâmetros de entrada utilizados como padrão, são eles:

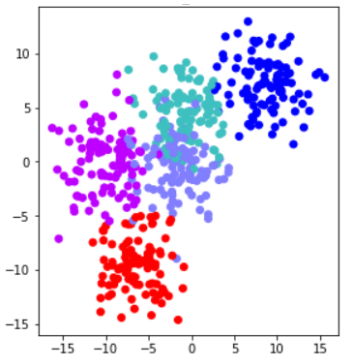


Figura 17 – Visualização de base de dados gerada (Do autor).

- n_features (número de classes para cada objeto): 2
- n_samples (número de amostras): 500
- centers (número de grupos): 5
- cluster_std (sobreposição - o quanto os pontos de cada grupo vão se sobrepor): 2.5

Os parâmetros restantes não foram especificados e, portanto, possuem valor padrão. A Figura 17 ilustra a visualização dessa base de dados gerada com os valores especificados acima.

Para avaliar o desempenho de cada algoritmo, é preciso executá-lo em diferentes bases de dados. Para isso, foram gerados dados variando individualmente os parâmetros descritos a cima: **n_samples**, **centers** e **cluster_std** em diversos intervalos. Quando um parâmetro estiver variável, os outros valores permanecem constantes e assumem os valores padrão descritos acima. Além disso, os algoritmos foram executados dez vezes em cada combinação de parâmetros, de forma a calcular a média e o desvio padrão dos resultados obtidos a partir deles.

A tabela 1 demonstra os intervalos de variação para cada um dos parâmetros, sendo "passo" a diferença entre duas variações consecutivas. As Figuras 18, 19 e 20 ilustram alguns exemplos das bases de dados geradas para algumas variações de parâmetros.

Tabela 1 – Intervalo de variação de parâmetros para geração dos dados

Parâmetro	**Intervalo**	**Passo**	**Total de bases de dados**
Sobreposição	0 - 25	1	26
Número de grupos	1 - 100	5	21
Número de amostras	100 - 1000	50	19

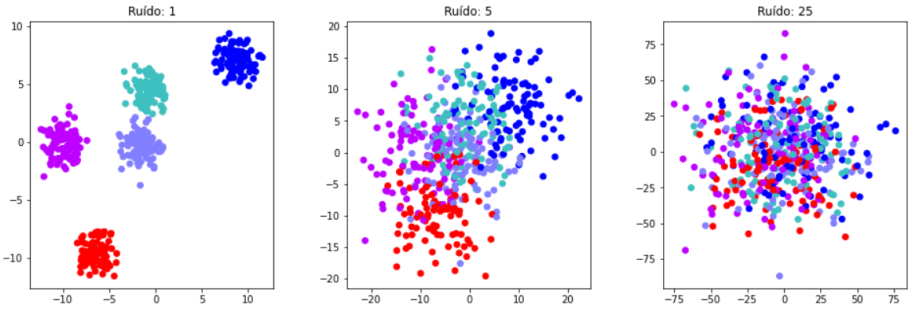


Figura 18 – Geração de dados: Variação da sobreposição (Do autor).

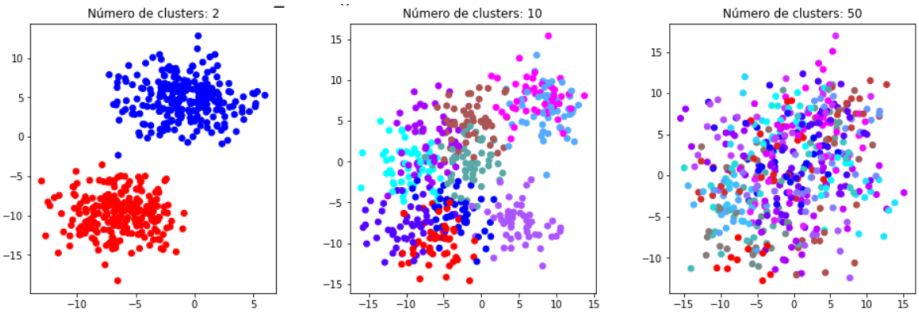


Figura 19 – Geração de dados: Variação do número de grupos (Do autor).

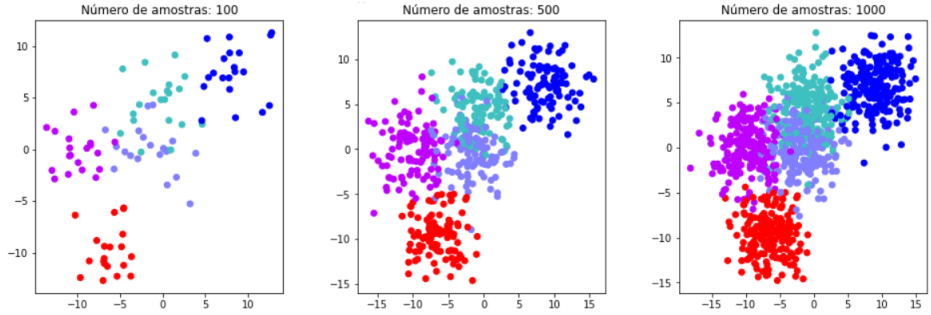
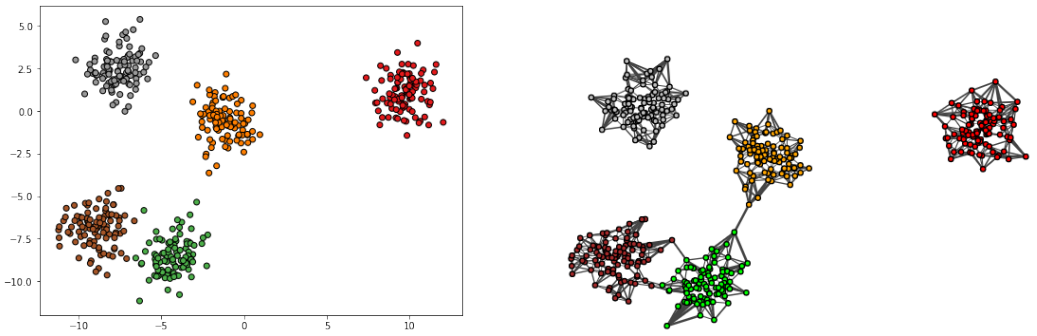


Figura 20 – Geração de dados: Variação do número de amostras (Do autor).

3.4 Construção de Redes

Os algoritmos de Detecção de Comunidades são executados baseados em dados organizados no formato de redes e, por isso, é necessária a utilização de algoritmos que recebam os dados gerados e construam as redes que serão utilizadas. Os algoritmos selecionados para isso foram KNN, Mutual KNN e E-Vizinhança

As Figuras 21(a) e 21(a) ilustram, respectivamente, uma base de dados gerada artificialmente e sua respectiva rede construída no algoritmo KNN com k=7.



(a) (b)

Figura 21 – Base de dados gerada artificialmente: (a) ilustra a base de dados desenhada em duas dimensões; (b) ilustra a rede construída pelo algoritmo KNN (k=7) com base nos dados gerados artificialmente (Do autor).

3.5 Teste estatístico

No geral, não há um procedimento estabelecido para comparar o desempenho de classificadores em vários conjuntos de dados. Existem diferentes técnicas estatísticas que podem ser adotadas e que dependem da experiência para decidir se as diferenças entre os algoritmos são reais ou aleatórias.

Demšar define algumas diretrizes para realização desses testes estatísticos, por exemplo, o teste t-pareado, *Wincoxons's rank test*, *ANOVA*, *Friedman's test*, entre outros. Para usuários não experientes em testes estatísticos pode ser um grande desafio escolher a diretriz correta, uma vez que não é uma decisão intuitiva ou trivial [10].

Para realizar uma comparação estatística das medidas de desempenho obtidas pelos algoritmos dentre diferentes bases de dados utilizadas deste trabalho, foi utilizada a biblioteca Autorank (https://github.com/sherbold/autorank) [19] do Python. O objetivo do Autorank é simplificar a análise estatística para não especialistas pois uma única chamada de função lida com a escolha das diretrizes citadas acima. Funções adicionais permitem a geração de gráficos apropriados, tabelas de resultados e até mesmo um documento completo de látex. A entrada necessária para o autorank são os dados sobre as populações em um dataframe do *Pandas*.

O autorank utiliza a seguinte abordagem para realizar a comparação estatística: Primeiro, usa-se o teste de Shapiro-Wilk para verificar se a população dos resultados segue uma distribuição normal e é utilizada a Correção de Bonferoni para esses testes. Caso os dados sigam uma distribuição normal, é usado então o Teste de Bartlett para homogeneidade, caso contrário é utilizado o Teste de Levene. Por fim, baseado nos testes de normalidade e homogeneidade, o autorank seleciona os testes e métodos apropriados para determinar os intervalos de confiança para comparação estatística. Este intervalo

de confiança é representado nos experimentos como “CD”, considerada a distância crítica para que os algoritmos possam ser considerados estatisticamente semelhantes.

3.6 Experimentos

Os experimentos foram realizados em linguagem python utilizando as bibliotecas já citadas anteriormente: scikit-learn e igraph. A estrutura do código pode ser dividida em 3 níveis de repetição: no primeiro nível, foram divididos os blocos de parâmetro variado (sobreposição, número de grupos e número de amostras); no segundo nível, dentro de cada bloco de parâmetro variado, cada algoritmo foi executado 10 vezes em diferentes bases de dados (mantendo suas respectivas características) calculando a média e desvio padrão do desempenho de cada algoritmo; o último nível é composto por blocos de repetição para algoritmos que necessitam da definição de algum parâmetro, em que os algoritmos foram executados dentro de um intervalo de valores para cada um dos parâmetros e para cálculo da média do algoritmo para aquela base de dados foi considerado somente o parâmetro cujo algoritmo obteve melhor desempenho, os intervalos de variação para cada parâmetro está descrito na tabela 2, sendo “passo” o tamanho da variação de cada iteração no intervalo.

Tabela 2 – Intervalo de variação para definição ideal de parâmetros

Algoritmo	**Parâmetro**	**Intervalo**	**Passo**
k-NN	k	1 - 40	1
Mk-NN	k	1 - 40	1
E-Vizinhança	ϵ	0.25 - 10.0	0.25
DBSCAN	EPS	0.1 - 8.0	0.20

Por fim, considerando o impacto da construção da rede para execução dos algoritmos de Detecção de Comunidades, foi realizada uma combinação entre os algoritmos de Geração de Redes e Detecção de Comunidades, garantindo que o impacto do resultado do agrupamento seja explicitado.

A próxima seção apresenta uma análise dos resultados obtidos no experimento.

4 Análise e discussão dos resultados

As seções deste capítulo mostram os resultados experimentais da execução dos algoritmos selecionados em diferentes conjuntos de base de dados. A análise dos resultados se dá em 4 partes: resultado dos algoritmos aplicados em bases de dados variando a sobreposição, variando o número de amostras, variando o número de grupos e, por fim, um experimento para entender o comportamento dos parâmetros dos algoritmos de construção de redes com a variação do número de grupos.

4.1 Variando a sobreposição

A Figura 22 demonstra o desempenho dos algoritmos na variação da sobreposição utilizando o NMI como parâmetro de comparação.

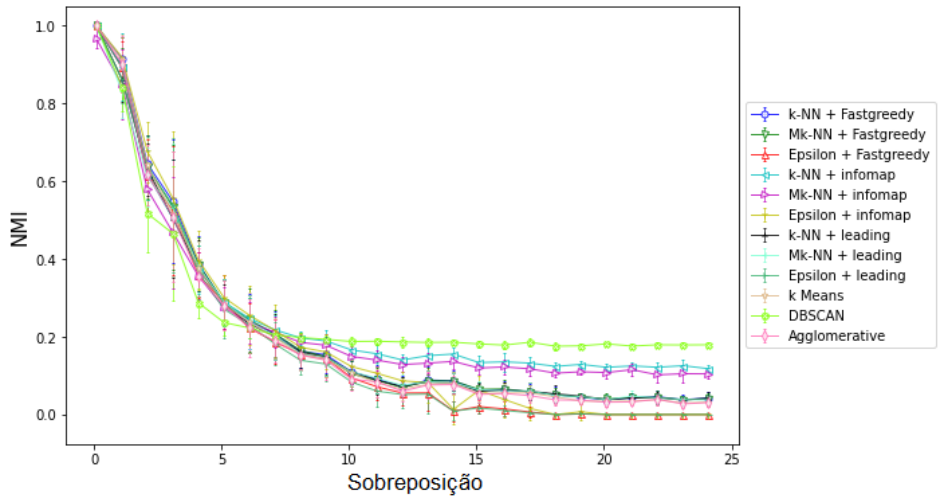


Figura 22 – Desempenho dos algoritmos variando sobreposição (Do autor).

A Tabela 3 foi construída com a média e desvio padrão dos NMIs obtidos por cada algoritmo.

Por fim, a Figura 23 representa o resultado gerado ao rodar a análise estatística nos resultados obtidos. A análise estatística foi realizada para 12 populações com 25 amostras pareadas. O autorank identificou que a distribuição das populações não é normal e, por isso, foi utilizado o teste não-paramétrico de Friedman para determinar se existem diferenças significativas entre os valores medianos das populações. Foi utilizado o teste Nemenyi para inferir quais diferenças são significativas. As diferenças entre as populações são significativas se a diferença da classificação média for maior do que a distância crítica CD = 3,333 do teste de Nemenyi.

Tabela 3 – NMI Médio e Desvio Padrão variando a sobreposição

Algoritmo	**Média**	**Desvio Padrão**
k-NN + Infomap	0,3616146047	0,2849660059
DBSCAN	0,3526382669	0,2629490708
Mk-NN + Infomap	0,3404994286	0,2750181413
Epsilon + Infomap	0,3391288578	0,3234066489
k-NN + Fastgreedy	0,3385524071	0,3149280441
Mk-NN + Fastgreedy	0,3353015752	0,3112255197
kMeans	0,3305353447	0,3158354638
k-NN + Leading Eigenvector	0,328466171	0,3060191297
Mk-NN + Leading Eigenvector	0,3274192882	0,301503819
Agglomerative	0,3233451123	0,3141854041
Epsilon + Fastgreedy	0,3148650159	0,3219961499
Epsilon + Leading Eigenvector	0,3111552544	0,3220247468

Desse gráfico, é possível concluir que, no geral, os algoritmos de Detecção de Comunidades tiveram um desempenho melhor que dos algoritmos de Agrupamento, com exceção do DBSCAN, que foi o segundo algoritmo com melhor desempenho.

Sendo "CD" a distância máxima em que dois algoritmos podem ser considerados semelhantes estatisticamente, é possível constatar que, apesar de existirem alguns grupos de algoritmos sem diferença estatisticamente significativa, há um grupo de algoritmos que desempenhou melhor (formado por k-NN + infomap, DBSCAN, k-NN + Fastgreedy, Mk-NN + informap, Epsilon + infomap e Mk-NN + leading) e um grupo com o pior desempenho (formado por Epsilon + leading, Epsilon + Fastgreedy, Agglomerative, e kMeans).

Além disso, foi possível constatar também que o algoritmo de construção de redes adotado impactou nos resultados dos algoritmos de detecção de comunidades, uma vez que houve diferença estatisticamente significativa entre Epsilon + Fastgreedy e K-NN + Fastgreedy/Mk-NN + Fastgreedy.

Comparando a tabela de médias e desvios padrão e o gráfico de comparação estatística, nota-se que os algoritmos estão ranqueados de forma semelhante, comprovando a ordem de desempenho. Outro ponto importante a se destacar é que os algoritmos com maior pontuação média tendem a ter um desvio padrão menor, evidenciando a maior precisão do agrupamento.

4.2 Variando o número de amostras

A Figura 24 demonstra o desempenho dos algoritmos na variação do número de amostras utilizando o NMI como parâmetro de comparação.

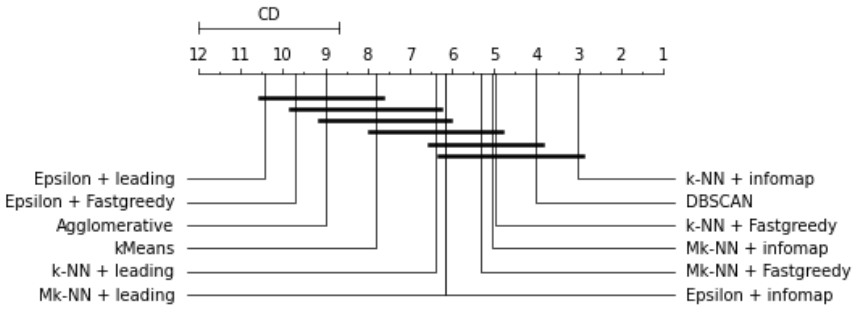


Figura 23 – Comparação estatística dos resultados variando sobreposição geradas pelo Autorank (Do autor).

Vale destacar que, apesar da quantidade de picos e vales presentes no gráfico, é possível observar que o desvio padrão de cada algoritmo tende a transformar as curvas em uma constante, demonstrando que o desempenho dos algoritmos variando o numero de amostras tende a se manter "constante".

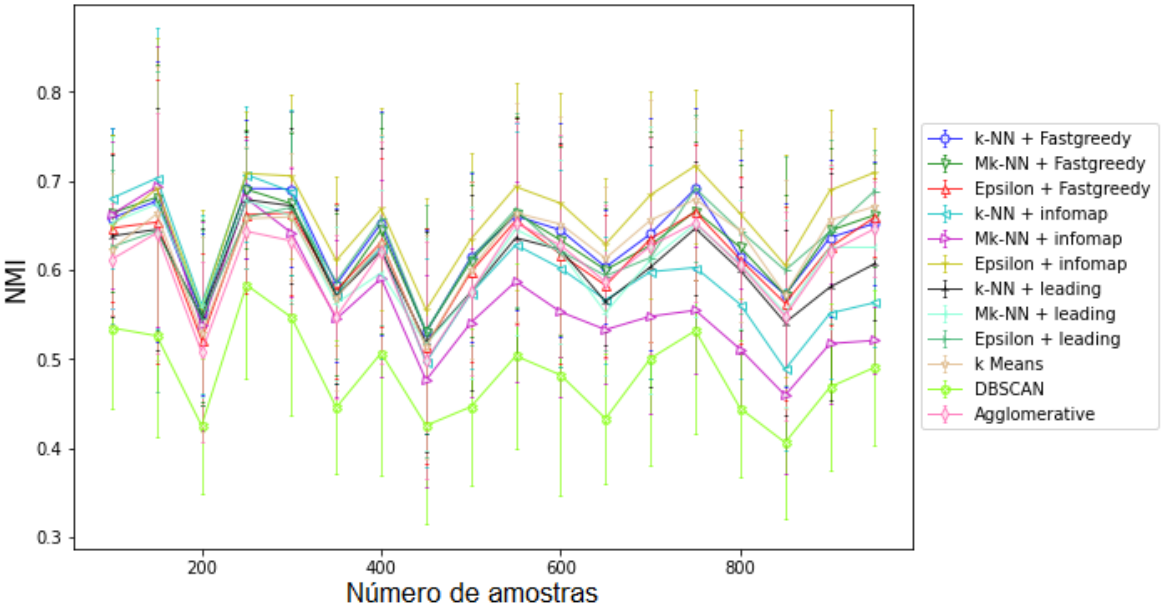


Figura 24 – Desempenho dos algoritmos variando número de amostras (Do autor).

A Tabela 4 foi construída com a média e desvio padrão dos NMIs obtidos em cada algoritmo.

Por fim, a Figura 25 representa o resultado gerado ao rodar a análise estatística nos resultados obtidos. A análise estatística foi realizada para 12 populações com 18 amostras pareadas. O autorank identificou que a distribuição de todas as populações é normal. Nesse caso, apesar da interpretação do resultado ser feita da mesma maneira que anteriormente, o gráfico foi representado de uma forma diferente pois os resultados obtidos

Tabela 4 – NMI Médio e Desvio Padrão variando o número de amostras

Algoritmo	**Média**	**Desvio Padrão**
Epsilon + Infomap	0,6523597816	0,0514352237
k-NN + Fastgreedy	0,6334044334	0,05146146451
Mk-NN + Fastgreedy	0,6304509773	0,05032300421
Epsilon + Leading Eigenvector	0,6217381306	0,04412288585
k-NN + Infomap	0,6195274082	0,06585914127
kMeans	0,6182619308	0,05047361111
Epsilon + Fastgreedy	0,6147337355	0,0503757669
Mk-NN + Leading Eigenvector	0,6138551658	0,05081677286
k-NN + Leading Eigenvector	0,610640511	0,04798070376
Agglomerative	0,5982380244	0,05034691183
Mk-NN + Infomap	0,5892898338	0,06905176816
DBSCAN	0,4928253132	0,05175989029

seguem uma distribuição normal. Foi aplicado o teste de Bartlett para homogeneidade, constatando que os dados são homocedásticos. Por conta dessas características, foi utilizado o teste de ANOVA para determinar se há diferenças significativas entre os valores médios das populações, e então foi utilizado o teste Tukey HSD para inferir quais diferenças são significativas.

Para avaliar quais algoritmos são semelhantes estatisticamente, basta traçar uma reta vertical. Todos os algoritmos cortados por essa reta fazem parte de um conjunto sem diferença estatisticamente significativa. O traço vermelho na figura demonstra essa representação.

Pelo gráfico, é possível perceber que os algoritmos Epsilon + infomap tiveram o melhor desempenho. Além deles, com exceção do Mk-NN + infomap e DBSCAN, o resultado de todos os algoritmos restantes podem ser considerados semelhantes estatisticamente.

É válido notar que o algoritmo DBSCAN teve um desempenho inferior em relação aos outros, isso acontece pois o parâmetro EPS se mostrou muito sensível ao número de amostras quando fixando o número de grupos. Por conta disso, o algoritmo teve dificuldades de encontrar um valor ótimo de EPS no intervalo definido, afetando negativamente seu desempenho.

Novamente, o gráfico de comparação estatística e a tabela de médias possuem um ranking semelhante, evidenciando a diferença de desempenho entre os algoritmos.

4.3 Variando número de grupos

A Figura 26 demonstra o desempenho dos algoritmos na variação do número de grupos utilizando o NMI como parâmetro de comparação.

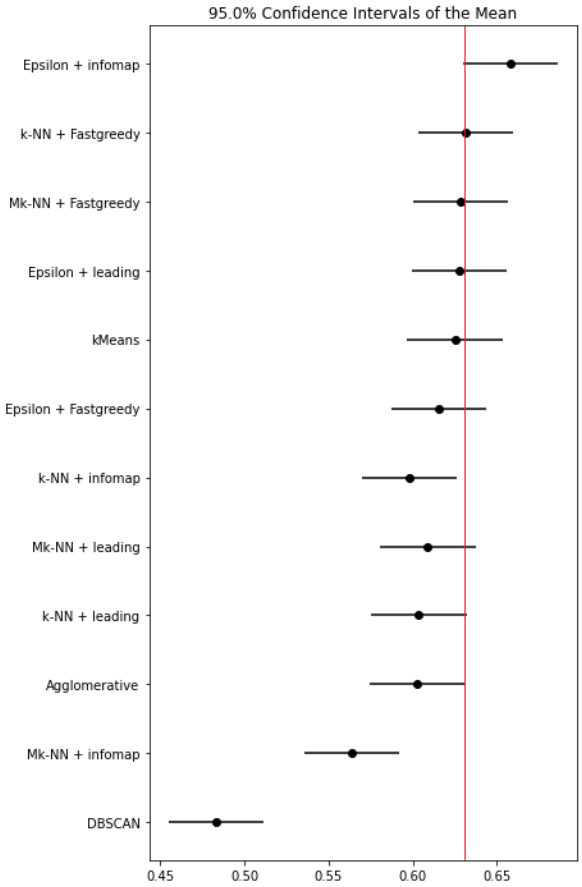


Figura 25 – Comparação estatística dos resultados variando número de amostras geradas pelo Autorank com traço exemplificando grupo de algoritmos semelhantes estatisticamente (Do autor).

Pela figura é possível perceber que o algoritmo Infomap não é capaz de realizar a classificação para um número baixo de grupos. Provavelmente, isso ocorre por conta da estratégia multinível utilizada, que cria uma rede contraída mesmo a rede possuindo 1 ou poucos grupos.

A Tabela 5 foi construída com a média e desvio padrão dos NMIs obtidos em cada algoritmo.

Por fim, a Figura 27 representa o resultado gerado ao rodar a análise estatística nos resultados obtidos. A análise estatística foi realizada para 12 populações com 20 amostras pareadas. O autorank identificou que a distribuição das populações não é normal e, por isso, foi utilizado o teste não-paramétrico de Friedman para determinar se existem diferenças significativas entre os valores medianos das populações. Foi utilizado o teste Nemenyi para inferir quais diferenças são significativas. As diferenças entre as populações são significativas se a diferença da classificação média for maior do que a distância crítica CD = 3,726 do teste de Nemenyi.

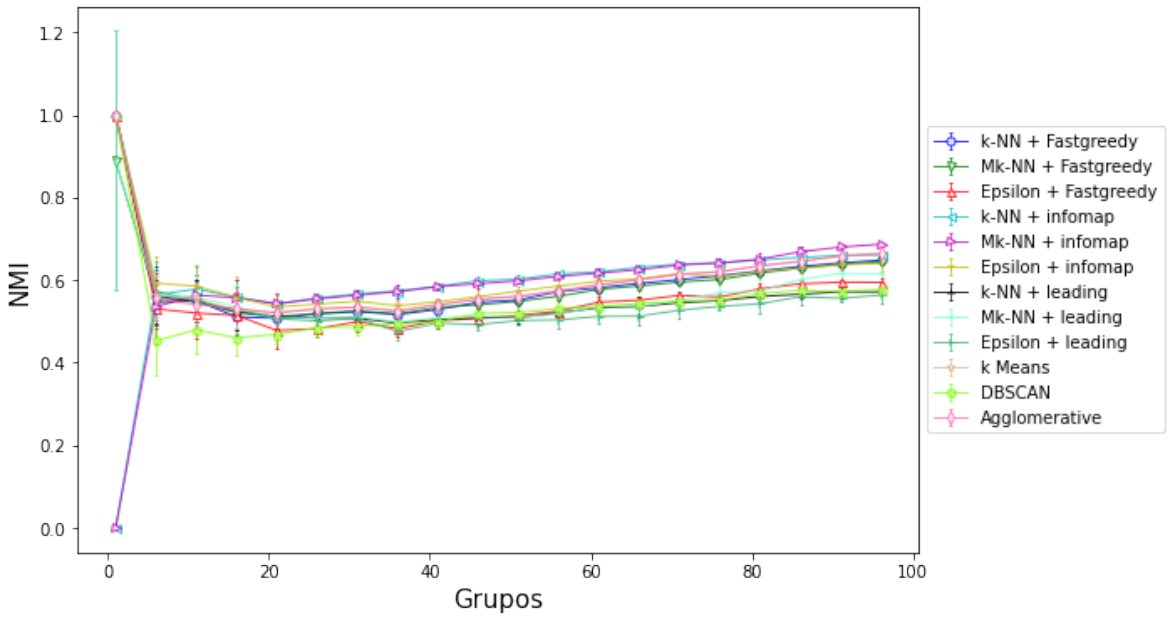


Figura 26 – Desempenho dos algoritmos variando número de grupos (Do autor).

Tabela 5 – NMI Médio e Desvio Padrão variando o número de grupos

Algoritmo	**Média**	**Desvio Padrão**
Epsilon + Infomap	0,6111987943	0,1288582403
kMeans	0,5990790609	0,1338830921
Agglomerative	0,5968235109	0,1346734479
k-NN + Fastgreedy	0,590280243	0,1369414006
Mk-NN + Fastgreedy	0,5790361804	0,1047053075
k-NN + Leading Eigenvector	0,5698565049	0,1408623126
Mk-NN + Leading Eigenvector	0,5656281847	0,1081356855
Epsilon + Leading Eigenvector	0,5647702693	0,143578695
Epsilon + Fastgreedy	0,562119703	0,14467531
DBSCAN	0,54582067	0,1508406599
k-NN + Infomap	0,529222348	0,1742720686
Mk-NN + Infomap	0,5240349195	0,1733356755

Sendo “CD” a distância máxima em que dois algoritmos podem ser considerados semelhantes estatisticamente, é possível constatar que, apesar de existirem alguns grupos de algoritmos semelhantes estatísticamente, há um grupo de algoritmos que desempenhou melhor (formado por k-NN + infomap, Mk-NN + infomap, Epsilon + infomap, kMeans e Agglomerative) e um grupo com o pior desempenho (formado por Epsilon + leading, DBSCAN, k-NN + leading, Epsilon + Fastgreedy, Mk-NN + leading, Mk-NN + Fastgreedy).

Com isso, é possível concluir que, nessa situação, o algoritmo com melhor desempenho foi o Infomap, independente do algoritmo de construção de rede associado. Com

exceção do DBSCAN, os algoritmos de agrupamento também desempenharam bem, não possuindo diferença estatisticamente significativa com o Infomap.

Por fim, observa-se também que nesse caso a escolha do algoritmo de construção de rede não gerou diferença estatisticamente significativa entre os algoritmos de detecção de comunidades.

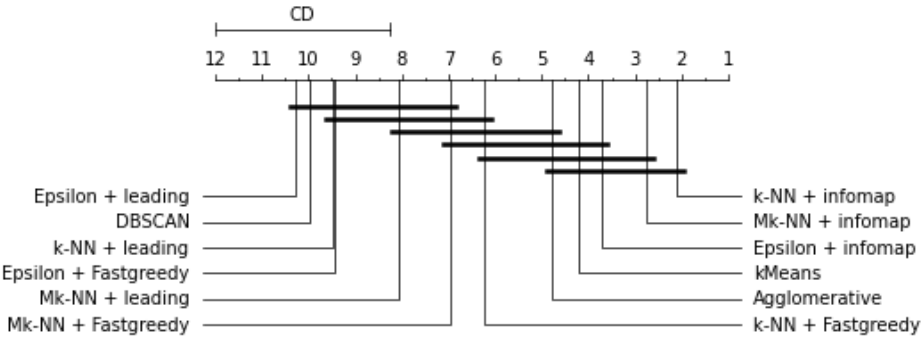


Figura 27 – Comparação estatística dos resultados variando número de grupos geradas pelo Autorank (Do autor).

Como aconteceu nos cenários anteriores, o gráfico gerado pelo Autorank e a tabela com as médias e desvios padrão possuem ranking similar, comprovando a diferença de desempenho entre os algoritmos.

4.4 Sensibilidade dos parâmetros nos algoritmos de construção de redes em relação ao número de grupos

Foi realizado um experimento para avaliar o comportamento dos parâmetros dos algoritmos de construção de redes na variação do número de grupos. Os parâmetros estudados são ϵ para E-Vizinhança e k para k-NN e Mk-NN. O algoritmo de detecção de comunidade escolhido para cálculo do NMI foi o Fastgreedy.

A Figura 28 mostra os resultados obtidos para k-NN + Fastgreedy variando o número de grupos e o parâmetro k entre 1 e 19. É possível perceber que, para um número de grupos pequeno em relação a base de dados, o k ideal deve ser um valor alto (entre 3 e 10). Em contrapartida, para uma situação de muitos grupos em relação a base de dados, o k ideal deve ser um valor baixo (entre 1 e 2).

Isso faz sentido pois, considerando um cenário com poucos grupos, seria necessário um alto número de k vizinhos para obter poucos grupos como resultado do agrupamento. Em contrapartida, caso haja muitos grupos, o ideal é que cada centróide tenha um baixo número de k vizinhos, visto que os grupos serão formados por poucos vértices.

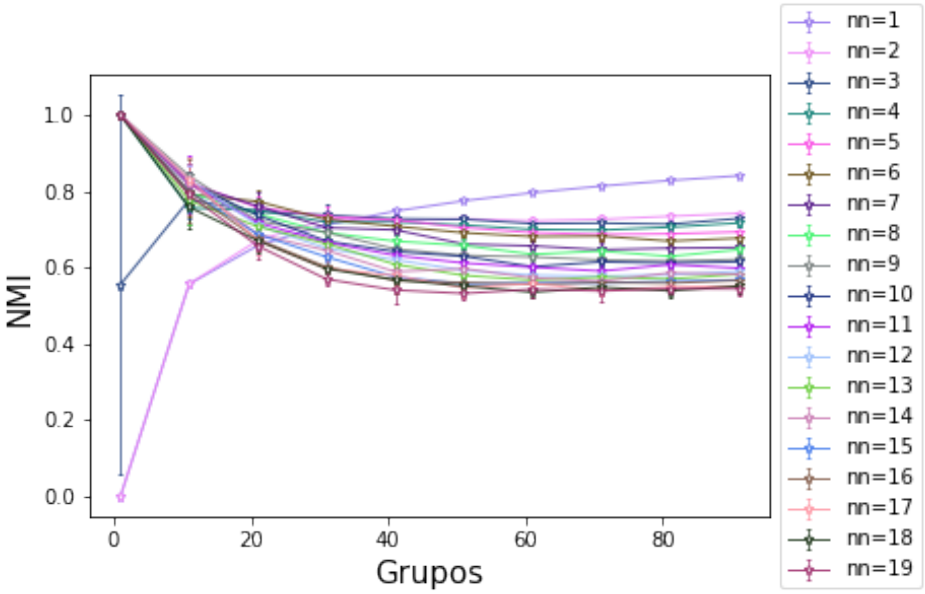


Figura 28 – Desempenho de k-NN + Fastgreedy variando número de grupos e parâmetro k (Do autor).

A Figura 29 mostra os resultados obtidos para o Mk-NN + Fastgreedy variando o número de grupos e o parâmetro k entre 1 e 19. É possível perceber uma tendência similar ao citado anteriormente, destacando que o algoritmo Mk-NN não desempenha bem em situações com poucos grupos em relação a base de dados.

Esse comportamento faz sentido, pois considerando um cenário com poucos grupos em relação a base de dados o algoritmo Mk-NN teria dificuldade para realizar o agrupamento uma vez que, para formar um grupo, a regra de k vizinhos deve ser satisfeita mais vezes, aumentando a dificuldade de agrupamento.

Por fim, a Figura 30 mostra os resultados obtidos para o E-Vizinhança + Fastgreedy variando o número de grupos e o parâmetro ϵ entre 0.5 e 9.5. É possível perceber um comportamento similar aos citados anteriormente, porém é importante ressaltar a diferença de escala para os valores dos parâmetros e o baixo desempenho para valores altos de ϵ para quantidades intermediárias de número de grupos. Isso significa que, no geral, o valor de ϵ ideal tende a ser baixo (abaixo de 3), reduzindo conforme o número de grupos aumenta.

Este comportamento também faz sentido, uma vez que em um cenário com poucos grupos, é necessário um ϵ grande para que a circunferência dos grupos seja maior de forma a incluir um número maior de amostras em cada grupo.

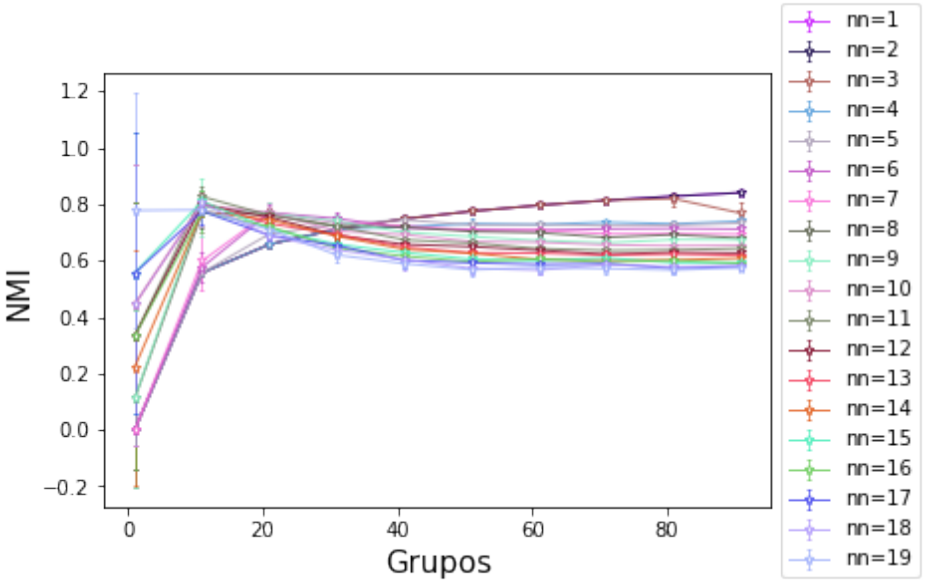


Figura 29 – Desempenho de Mk-NN + Fastgreedy variando número de grupos e parâmetro k (Do autor).

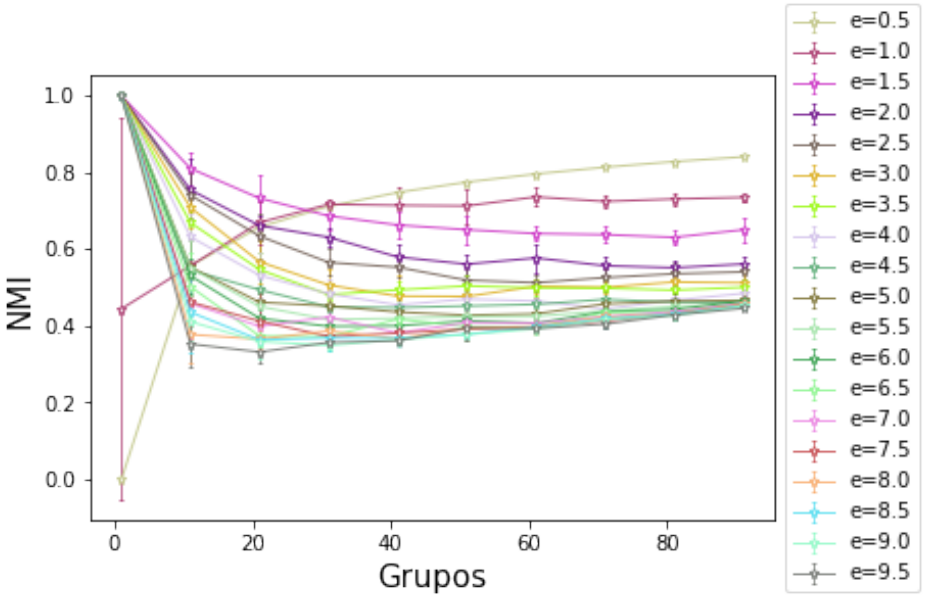


Figura 30 – Desempenho de E-Vizinhanca + Fastgreedy variando número de grupos e parâmetro ϵ (Do autor).

5 Conclusão

Este trabalho compara os métodos de Agrupamento e Detecção de comunidades em conjuntos de bases de dados geradas artificialmente. De modo geral, é possível observar que a hipótese definida inicialmente está incorreta, uma vez que, a partir dos experimentos, é possível concluir que os algoritmos possuem diferença estatisticamente significativa em relação a medida de desempenho adotada.

Além disso, observa-se que, em alguns casos, a escolha do algoritmo de construção de redes para execução dos métodos de detecção de comunidades pode impactar no desempenho do agrupamento. Esse comportamento pode ser observado nos experimentos em que há variação da sobreposição. De modo geral, o algoritmo Infomap foi o que obteve melhor desempenho em todos os casos.

É válido ressaltar que a distorção de desempenho observada em algumas situações por conta da dificuldade em determinar o valor ideal de determinados parâmetros. Em especial houve grande dificuldade com a definição dos parâmetros do algoritmo DBSCAN, uma vez que é extremamente complicado determinar de forma precisa ambos os parâmetros na grande quantidade de base de dados em que os experimentos foram realizados. Uma possível solução para esse problema seria utilizar o algoritmo HDBSCAN, uma variação do DBSCAN que realiza um cálculo automático para os parâmetros [37].

Por fim, considerando o resultado dos experimentos, é possível concluir que, no geral, os algoritmos de detecção de comunidade são uma alternativa viável e em algumas situações até melhores para problemas que envolvem agrupamento de dados quando a métrica de interesse é o desempenho do agrupamento, em especial o NMI.

5.1 Trabalhos futuros

Para continuação deste trabalho ficam definidas algumas possibilidades que poderiam se abordadas a fim de melhorar o estudo e os resultados obtidos:

- Utilizar outras *bibliotecas* para geração dos dados, como *make_circles* e *make_moons*, realizando os experimentos em bases com diversos tipos de distribuição de dados.
- Ampliar a base de dados utilizadas nos experimentos, como dados reais ou com múltiplas dimensões.
- Fazer uso do método de Análise de Componentes Principais (PCA) para analisar bases de dados com múltiplas dimensões.

- Avaliar performance (velocidade) dos algoritmos utilizados nos experimentos.
- Utilizar a biblioteca *GridSearch* do scikit-learn para otimizar a definição dos parâmetros necessários para execução dos algoritmos.

Referências

[1] Arthur, David, and Sergei Vassilvitskii. *k-means++: The Advantages of Careful Seeding*. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics (2007) Citado na página 17.

[2] ARRUDA, G. F. de; COSTA, L. da F.; RODRIGUES, F. A. *A complex networks approach for data clustering.* Physica A: Statistical Mechanics and its Applications, Elsevier, v. 391, n. 23, p. 6174Ű6183, 2012. Citado na página 33.

[3] BARBARA, D. *An introduction to cluster analysis for data mining.* Retrieved November, v. 12, p. 2003, 2000. Citado na página 24.

[4] Batista, Gustavo Enrique. *Pré-processamento de Dados em Aprenziado de Máquina Supervisionado*. Tese de Doutorado - Universidade de São Paulo. São Carlos, 2003. Nenhuma citação no texto.

[5] BELKIN, M.; NIYOGI, P. *Laplacian eigenmaps for dimensionality reduction and data representation.* Neural Computation, v. 15, n. 6, páginas 1373–1396, 2003 Citado na página 33.

[6] BRITO, M. R.; CHAVEZ, E. L.; QUIROZ, A. J.; YUKICH, J. E. *Connectivity of the mutual k-nearest neighbor graph in clustering and outlier detection.* Statistics and Probability Letters, v. 35, n. 1, páginas 33–42, 1997. Citado 2 vezes nas páginas 11 e 32.

[7] Clauset, A., Newman, M. E. J., e Moore, C. (2004). *Finding community structure in very large networks.* Physical Review E, 70:066111. Citado na página 34.

[8] Datanovia. Hierarchical Clustering in R: The Essentials. Acessado em 2 de novembro de 2021, <https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/> Citado 2 vezes nas páginas 11 e 29.

[9] DAUMÉ III, H. *A course in machine learning.* Publisher, ciml. info, v. 5, p. 69, 2012. Citado na página 23.

[10] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, 7, 1-30. Citado na página 45.

[11] ESTER, M. et al. *Density-based spatial clustering of applications with noise. In: Int. Conf. Knowledge Discovery and Data Mining.* [S.l.: s.n.], 1996. v. 240. Citado 2 vezes nas páginas 17 e 27.

[12] EVERITT, B. S.; DUNN, G. *Applied multivariate data analysis.* [S.l.]: Wiley Online Library, 2001. v. 2. Citado na página 23.

[13] Freitas, Lusmar Mendes. *Uma Análise Comparativa entre Técnicas de Detecção de Comunidades com Aplicação para Problema de Agrupamento de Objetos Invariantes.* Trabalho de Conclusão de Curso - Universidade Federal de Uberlândia. Monte Carmelo, 2018. Citado na página 20.

[14] FORTUNATO, S. *Community detection in graphs.* Physics reports, Elsevier, v. 486, n. 3, p. 75Ű174, 2010. Citado 2 vezes nas páginas 33 e 34.

[15] GOWDA, K.; KRISHNA, G. *Agglomerative clustering using the concept of mutual nearest neighborhood.* Pattern Recognition, v. 10, n. 2, páginas 105–112, 1978 Citado na página 32.

[16] GUIDO, S.; MULLER, A. C. *Introduction to Machine Learning with Python: a guide for data scientists.* Rio de Janeiro: O'reilly, 2016. Citado na página 22.

[17] HAYKIN S. *Neural Networks - A Compreensive Foundation.* Prentice-Hall, Nova Jersey, segunda edição, 1999. Citado na página 22.

[18] HARPER, D. et al. *Online etymology dictionary.* 2001. Citado na página 17.

[19] Herbold, Steffen. *Autorank: A Python package for automated ranking of classifiers.* Journal of Open Source Software, v. 5, 2020. Citado na página 45.

[20] JAIN, A. K.; DUBES, R. *Algorithms for Clustering Data.* Prentice Hall. 1988 Citado na página 17.

[21] JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. *Data clustering: a review. ACM computing surveys (CSUR)*, Acm, v. 31, n. 3, p. 264–323, 1999. Citado 3 vezes nas páginas 23, 24 e 25.

[22] KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. *Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering*, Amsterdam, v. 160, n. 1, p. 3–24, 2007. Citado na página 22.

[23] KULTZAK, Adriano Francisco. *Classificação de textos utilizando algoritmos de aprendizagem de máquina com WEKA.* 2016. 74 f. Trabalho de Conclusão de Curso -

Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016. Nenhuma citação no texto.

[24] Berton, L. *Construção de redes baseadas em vizinhança para o aprendizado supervisionado*. 75f. Tese (Doutorado), Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2016. Citado 3 vezes nas páginas 11, 32 e 33.

[25] MAIER, M.; HEIN, M.; LUXBURG, U. *Cluster identification in nearest-neighbor graphs*. Em: ALT'07: Proceedings of the 18th International Conference on Algorithmic Learning Theory, 2007, páginas 196–210 Citado na página 32.

[26] Mendes, Jakelson Carreiro. *Agrupamento de Dados e suas Aplicações*. Trabalho de Conclusão de Curso - Universidade Federal do Maranhão. São Luís, 2017. 52 p. Citado 4 vezes nas páginas 11, 24, 26 e 28.

[27] Mitchell, 1997 - Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill. 12, 97 Citado na página 21.

[28] Monard, M. C., Batista, G. E. A. P. A., Kawamoto, S. & Pugliesi, J. B. (1997). *Uma Introdução ao Aprendizado Simbólico de Máquina.* Citado na página 21.

[29] Nadia Rahmah and Imas Sukaesih Sitanggang 2016. *Determination of Optimal Epsilon (EPS) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra*. IOP Conf. Ser.: Earth Environ. Sci. 31 012012 Citado 2 vezes nas páginas 28 e 29.

[30] NEWMAN, M. E.; GIRVAN, M. *Finding and evaluating community structure in networks.* Physical review E, APS, v. 69, n. 2, p. 026113, 2004. Citado 2 vezes nas páginas 34 e 36.

[31] Nielsen F. *Hierarchical Clustering. In: Introduction to HPC with MPI for Data Science*. Undergraduate Topics in Computer Science. Springer, Cham. 2016. Citado na página 30.

[32] Oliveira, Marcos de Souza . *Metodologia de seleção de features não supervisionada para clustering em conjunto de dados de alta dimensionalidade*. Dissertação de Pós Graduação - Universidade Federal de Pernambuco. Recife, 2018. 79 f.: il., fig., tab. Nenhuma citação no texto.

[33] PEDREGOSA, F. et al. *Scikit-learn: Machine learning in python.* the Journal of machine Learning research, JMLR. org, v. 12, p. 2825–2830, 2011. Citado na página 41.

[34] RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. *Near linear time algorithm to detect community structures in large-scale networks.* Physical review E, APS, v. 76, n. 3, p. 036106, 2007. Citado 2 vezes nas páginas 11 e 35.

[35] REZENDE, S. O.; PUGLIESI, J. B.; MELANDA, E. A.; PAULA, M. F., *Mineração de Dados*, in REZENDE, S. O. (Eds.), Sistemas Inteligentes, Editora Manole Ltda., p.307-335. 2003. Citado 3 vezes nas páginas 11, 21 e 22.

[36] Rosvall M., Axelsson D. e Bergstrom CT., *The map equation.* The European Physical Journal Special Topics, vol. 178, não. 1, pp. 13-23, 2009. Citado 3 vezes nas páginas 11, 35 e 37.

[37] R. Campello, D. Moulavi, and J. Sander, *Density-Based Clustering Based on Hierarchical Density Estimates* In: Advances in Knowledge Discovery and Data Mining, Springer, pp 160-172. 2013 Citado na página 57.

[38] Schmitt, Vinícius Fernandes. *Uma análise comparativa de técnicas de aprendizagem de máquina para prever a popularidade de postagens no facebook*. Trabalho de Conclusão de Curso - Universidade Federal do Rio Grande do Sul. Porto Alegre, 2018. Nenhuma citação no texto.

[39] SKINNER, B. F. *Are theories of learning necessary? Psychological review, American Psychological Association*, v. 57, n. 4, p. 193, 1950. Citado na página 17.

[40] SILVA, Bruno. *UTILIZAÇÃO DE APRENDIZAGEM DE MÁQUINA PARA CLASSIFICAÇÃO DE E-MAILS EM CATEGORIAS RELEVANTES.* 2021. Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Federal de São Carlos. São Carlos, 2021. Nenhuma citação no texto.

[41] SILVA, D. M.; BRITO, J. A. M.; OLIVEIRA, C. S. *Um Estudo Computacional Comparativo entre Algoritmos de Agrupamento e de Detecção de Comunidades*. In: SIMPÓSIO DE PESQUISA OPERACIONAL E LOGÍSTICA DA MARINHA, 19., 2019, Rio de Janeiro, RJ. Anais [...]. Rio de Janeiro: Centro de Análises de Sistemas Navais, 2019. Citado na página 20.

[42] STREHL, A.; GHOSH, J. *Cluster ensembles—a knowledge reuse framework for combining multiple partitions.* Journal of machine learning research, v. 3, n. Dec, p.583–617, 2002. Citado na página 39.

[43] TEIXEIRA, L. S.; LIMA, L. S.; ABREU, N. M. M. *Grafos que modelam redes confiáveis.* In: XL SBPO: A Pesquisa Operacional e o uso racional de recursos hídricos. João Pessoa, PB, Brasil, 2008 Citado na página 18.

[44] Valejo, A. (2014). *Refinamento multinível em redes complexas baseado em similaridade de vizinhança*. 75 f. Dissertação (Mestrado), Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Citado 5 vezes nas páginas 11, 31, 35, 36 e 38.

[45] XU, R.; WUNSCH, D. C. *Clustering. Hoboken.* [S.l.]: NJ: Wiley, 2009. Citado na página 23.

[46] ZHU, X. *Semi-supervised learning literature survey.* Relatório Técnico 1530, University of Wisconsin-Madison, computer Sciences, 2005. Citado na página 30.