UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

Vitor Freitas Xavier Soares

Análise de algoritmos de construção de grafos em conjuntos de dados de alta dimensionalidade

São Carlos - SP

2023

Vitor Freitas Xavier Soares

Análise de algoritmos de construção de grafos em conjuntos de dados de alta dimensionalidade

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

2023

Dedico este trabalho a todas as pessoas na minha vida que foram meu porto seguro quando eu estava perdido, que me fizeram rir quando estava apático, que me acolheram quando desisti de mim mesmo e me lembraram da felicidade por estar vivo.

Agradecimentos

Agradeço, imensamente, aos meus pais, Roseli e Jaime, pelo apoio incondicional, por todo o amor e paciência ao longo de todas as etapas da minha vida. Eles são os principais responsáveis pela minha formação acadêmica. Eu não tenho palavras para agradecer todo o esforço que vocês fizeram por mim ao longo de todos esses anos. Vocês são a razão pela qual eu pude participar de espaços que eu nunca imaginaria.

Agradeço a todos os meus amigos que foram fonte de inspiração, amor e felicidade. Escreveria milhões de qualidades sobre todos, porém vou falar de alguns que me marcaram durante o desenvolvimento deste trabalho. Agradeço à minha amiga Júlia Hansen que sempre acreditou em mim, quando eu mesmo não acreditava, por ser esse ser humano maravilhoso e de luz. Ao Gustavo Moreira agradeço por fazer parte da minha vida e ser uma pessoa incrível. Ao Jhonata, Ana Júlia e Amanda por terem sido as melhores companhias para dividir apartamento e por terem se tornado especiais na minha vida. Camilinha eu sou extremamente feliz por ter te conhecido. Você é luz por onde passa e quero viver diversos outros momentos especiais contigo. Gabriel Henrique obrigado por ter sido a pessoa amorosa e leve que conseguiu me animar até nos momentos mais difíceis - você é maravilhoso. Luis, você é muito especial para mim e sinceramente não há palavras para te agradecer, porém vale a tentativa. Obrigado por ter se tornado importante na minha vida tão rapidamente, por ter me aguentado nas oscilações constantes de humor nesse fim de graduação e por compartilhar as mais diversas e icônicas experiências. Sua amizade representa o mundo para mim. Às demais pessoas que não citei diretamente, porém que foram muito importantes durante toda a jornada, meus mais sinceros agradecimentos. Posso não ter falado de todos, porém vocês estão todos no meu coração e não sei nem como agradecer a amizade, as risadas e todo o amor que recebi durante todos esses anos.

Aos professores da UFSCar é necessário uma divisão para agradecer apropriadamente. Aos reais professores que nos encorajam e nos guiam através do conhecimento, agradeço o incentivo e por tornarem a jornada acadêmica fascinante. Aos demais agradeço por serem a prova viva que em nossa vida teremos diversas pedras no caminho, porém não há nada que nos impeça de seguir em frente a não ser nós mesmos.

Também agradeço imensamente ao professor Dr. Alan Demétrius Baria Valejo por ter me apresentado ao tema, pela orientação ao longo de todo trabalho e por todos os ensinamentos. Muito obrigado professor por toda a ajuda e orientação!

"Aqueles que passam por nós, não vão sós, não nos deixam sós. Deixam um pouco de si, levam um pouco de nós"

(Antoine de Saint-Exupéry)

Resumo

O rápido desenvolvimento e adoção de tecnologias de informação, como a internet, resultou em uma explosão na geração de dados, aumentando significativamente o volume de novas instâncias com diversos atributos. A vasta quantidade de atributos existentes em determinados conjuntos de dados não implica necessariamente em uma melhor performance, visto que o desempenho dos algoritmos diminui à medida que atributos são adicionados excessivamente (Maldição da Dimensionalidade). Assim, para analisar tais dados, no campo do aprendizado de máquina, os algoritmos de agrupamento tradicionais desempenham um papel crucial, segmentando-os com base em semelhanças. Uma abordagem mais recente é associar algoritmos de construção de grafos a algoritmos de detecção de comunidades. Nesta, os grafos representam as relações entre os dados, enquanto os algoritmos de detecção de comunidade ficam encarregados de revelar grupos densamente conectados e identificar padrões ocultos. Diante disso, o objetivo deste estudo é realizar uma análise de algoritmos de construção de grafos combinada com a detecção de comunidades em conjuntos de dados com alta dimensionalidade. Foram utilizados conjuntos de dados sintéticos com a métrica de avaliação principal sendo o índice externo denominado Informação Mútua Normalizada (NMI). Além disso, foram avaliados conjuntos de dados com diferentes quantidades de atributos, grupos, amostras e sobreposição, com o propósito de observar os efeitos da dimensionalidade nos resultados obtidos. Os algoritmos de agrupamento tradicionais *K-means* e *Agglomerative Clustering* foram considerados como *baseline* para a comparação do desempenho dos demais. Os resultados apontaram que algoritmos de construção de grafos associados aos algoritmos de detecção de comunidade são uma alternativa viável e apresentam até mesmo resultados melhores em alguns casos que os algoritmos de agrupamento tradicionais para conjuntos de dados com alta dimensionalidade.

Palavras-chave: Aprendizado da Máquina; Algoritmos de Agrupamento; Construção de Grafos; Detecção de Comunidade; Maldição da Dimensionalidade;

Abstract

The rapid development and adoption of information technologies, such as the internet, have resulted in an explosion in data generation, significantly increasing the volume of new instances with various attributes. The vast number of attributes present in certain datasets does not necessarily imply better performance, as algorithm performance decreases as attributes are excessively added (Curse of Dimensionality). Therefore, in the field of machine learning, traditional clustering algorithms play a crucial role in analyzing such data by segmenting it based on similarities. A more recent approach involves associating graph construction algorithms with community detection algorithms. In this approach, graphs represent the relationships between the data, while community detection algorithms are responsible for revealing densely connected groups and identifying hidden patterns. In light of this, the aim of this study is to perform an analysis of graph construction algorithms combined with community detection on high-dimensional datasets. Synthetic datasets were used, with the primary evaluation metric being the normalized mutual information index (NMI). Additionally, datasets with different numbers of attributes, groups, samples, and overlap were evaluated to observe the effects of dimensionality on the results. Traditional clustering algorithms, such as K-means and Agglomerative Clustering, were considered as baselines for comparing the performance of others. The results indicated that graph construction algorithms combined with community detection algorithms are a viable alternative and even yield better results in some cases than traditional clustering algorithms for high-dimensional datasets.

Keywords: Machine Learning; Traditional Clustering Algorithms; Graph Construction; Community Detection; Curse of Dimensionality.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A evolução tecnológica, a ampliação dos usuários com acesso a internet, o uso generalizado de dispositivos eletrônicos como celulares, computadores e relógios digitais, além das redes sociais, impulsionou a humanidade na geração de uma quantidade enorme de dados que são coletados a todo instante. Tais dados foram reconhecidos pelas principais empresas do mundo como um recurso essencial, visto que trazem *insights* e podem personalizar a experiência para os seus usuários de forma positiva. Com isso, cada ação em aplicativos, sites e redes sociais gera uma quantidade significativa de informações, as quais são armazenadas para serem analisadas. Dado esse contexto, surge um obstáculo: como tratar e processar essa gama de dados armazenados. Uma das principais áreas de pesquisa que estuda esse desafio é o Aprendizado de Máquina.

O Aprendizado de Máquina (AM), subárea da inteligência artificial (IA), tem como propósito principal a criação de algoritmos capazes de identificar padrões e tomar decisões com base em dados, sem a necessidade de programação explícita (MITCHELL, 1997). Dentro do AM é possível citar dois principais paradigmas de aprendizado: supervisionado e o não-supervisionado. No primeiro paradigma, os dados de treinamento são rotulados, permitindo que o modelo aprenda a fazer previsões com base nas respostas previamente conhecidas. Já no segundo, é necessário que o modelo busque padrões ou estruturas naturais nos dados. Uma área específica do aprendizado não-supervisionado é o agrupamento (ou *clusterização*). O agrupamento de dados tradicional consiste em agrupar objetos de acordo com determinado critério de similaridade ou medidas de proximidade. Os algoritmos tradicionais de agrupamento que podem ser mencionados são, por exemplo, *K-means* e o Agrupamento Aglomerativo.

Além dos algoritmos tradicionais, nos últimos tempos ganharam renome os algoritmos de agrupamento baseados em grafos. Grafos, em suma, são estruturas matemáticas que representam relações entre objetos, nas quais os nós representam os objetos em si e as arestas representam as relações entre esses objetos. A construção dos grafos requer o uso de algoritmos específicos, que levam em consideração os atributos e a estrutura entre os dados. Esses algoritmos podem ser baseados em similaridade, distâncias (como a Euclidiana, Manhattan, etc) ou outras métricas que façam sentido dado o problema a ser resolvido. Uma abordagem interessante é combinar algoritmos de construção de grafos com algoritmos de detecção de comunidades. Com isso, os algoritmos de construção de grafos transformam o conjunto de dados em um grafo, formando a estrutura necessária para a detecção de comunidades. Algoritmos de detecção, como o *Fastgreedy* e o *Leading Eigenvector*, são fundamentais para identificar grupos de nós altamente conectados, revelando estruturas e padrões latentes nos dados. Além disso, cabe destacar, que existem diferentes algoritmos

para transformar dados vetoriais em grafos, sendo que diferentes algoritmos de construção de grafos produzem topologias e padrões de conectividade diversas, o que gera distintos resultados, influenciando o desempenho dos algoritmos de detecção de comunidades.

Uma adversidade extra no AM é a alta dimensionalidade dos dados, que se refere ao grande número de atributos presentes nos conjuntos de dados (SIRIMONGKOLKASEM; DRIKVANDI, 2019). A alta dimensionalidade pode dificultar o processo de aprendizado, tendo em vista que a presença de muitos atributos irrelevantes ou redundantes pode prejudicar a performance dos algoritmos. Essa problemática é conhecida como a Maldição da Dimensionalidade, sendo contraintuitiva visto que mais dados com atributos relevantes acabam facilitando o aprimoramento dos modelos. Porém, o excesso deles sem pertinência acaba por interferir negativamente no desempenho final. Além disso, conforme o número de atributos aumenta, a tarefa de identificar padrões e relações torna-se mais complexa. Como consequência, podem surgir problemas como esparsidade dos dados, sobreposição de comunidades e dificuldades na detecção de agrupamentos relevantes.

Assim, surge a necessidade da existência de pesquisas que busquem analisar os métodos de construção de grafos com detecção de comunidades e avaliar se existe uma diferença significativa na precisão desses algoritmos quando comparados aos algoritmos tradicionais de agrupamento dado diferentes dimensionalidades. Entretanto, poucos estudos analisam a performance de tais algoritmos nesse cenário (YINGFAN; HONG; JIANGTAO, 2021), tornando uma lacuna de interesse para exploração científica.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo realizar uma análise de algoritmos de construção de grafos em conjunto de dados com alta dimensionalidade, principalmente, no contexto de detecção de comunidades. A hipótese que espera-se confirmar é de que algoritmos de agrupamento em grafos (ou detecção de comunidades) apresentam desempenho equiparável aos algoritmos tradicionais de agrupamento para tais dados ou superior, levando em conta a métrica de avaliação selecionada para a comparação entre os algoritmos e o conjunto de dados utilizados.

1.1.2 Objetivos específicos

Os objetivos específicos podem ser elencados em:

- Contribuir com a literatura sobre o desempenho dos algoritmos de construção de grafos em conjuntos de alta dimensionalidade;

- Gerar grafos a partir dos dados através de algoritmos de construção;
- Avaliar os algoritmos de detecção de comunidades executados sobre os grafos previamente gerados;
- Analisar os resultados obtidos, comparando os algoritmos de construção de grafos com detecção de comunidades com algoritmos tradicionais dadas diferentes dimensões.

1.2 Organização do trabalho

Este trabalho está dividido em cinco capítulos. No Capítulo 1, são apresentados a proposta de pesquisa, os objetivos e a justificativa para a realização do mesmo. O Capítulo 2 abrange a fundamentação teórica necessária para cumprir com o objetivo proposto, passando pelos conceitos de AM e dimensionalidade, além de explicações sobre os algoritmos utilizados e das métricas escolhidas para avaliar os resultados. No Capítulo 3, é exibida a metodologia utilizada para a execução dos experimentos. O Capítulo 4 destina-se a análise dos resultados obtidos, além da comparação entre os algoritmos e diferentes dimensionalidades. Por fim, no Capítulo 5, são expostas as conclusões do trabalho e possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os principais conceitos que embasam o presente trabalho. Sendo esses, aprendizado de máquina com foco nos algoritmos de agrupamento e detecção de comunidades, além do conceito de dimensionalidade.

2.1 Inteligência Artificial

O avanço da inteligência humana com o decorrer do processo de evolução tornou possível que o homo sapiens prevalecesse sobre o meio ambiente. A raça humana pode ser considerada como dotada de inteligência, principalmente, devido à capacidade de se adaptar e aprender de acordo com a situação e o meio (ERTEL, 2018).

A Inteligência Artificial (IA) veio como um campo que busca gerar máquinas capazes de simularem o comportamento humano. Assim, a IA pode ser vista como a habilidade de criar agentes inteligentes, os quais percebem seu ambiente e tomam decisões racionais para maximizar suas chances de sucesso (RUSSELL; NORVIG, 2016).

2.2 Aprendizado de Máquina

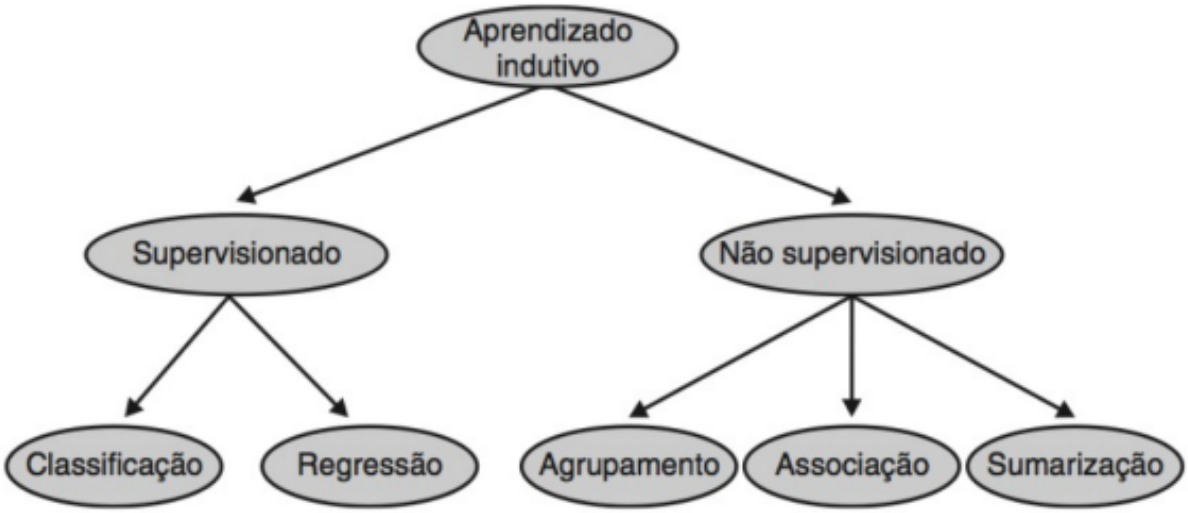
O Aprendizado de Máquina (AM) é uma subárea da IA que compreende métodos computacionais capazes de adquirir conhecimento de forma automática (MITCHELL, 1997). Por meio de métodos de AM, sistemas computacionais podem aprender conceitos e descobrir estruturas intrínsecas dos dados, transformando-os em informações úteis para a tomada de decisão em diversos ambientes. De acordo com a natureza dos dados é necessário utilizar diferentes metodologias e paradigmas de aprendizado para conseguir tratar corretamente os mesmos.

Dentro do AM tem-se uma abordagem muito utilizada: o aprendizado indutivo. Este é uma abordagem que tem base na inferência de regras ou padrões em relação a certos exemplos observados. O aprendizado indutivo tem como objetivo ser capaz de generalizar informações para dados não observados previamente a partir de exemplos de treinamento. O aprendizado indutivo é categorizado em duas abordagens principais: aprendizado supervisionado e não supervisionado conforme é ilustrado na Figura 1.

2.2.1 Aprendizagem supervisionada

No contexto do aprendizado supervisionado, o objetivo é desenvolver um modelo capaz de identificar padrões e comportamentos em dados de treinamento para prever

Figura 1 – Estrutura do aprendizado indutivo.



Fonte: (FACELI et al., 2011).

futuramente determinada informação (FACELI et al., 2011). Esses modelos preditivos são baseados em algoritmos supervisionados, uma vez que contam com uma "fonte externa" que fornece a classe desejada para cada exemplo. A classe representa o conceito a ser aprendido e pode ser classificado em duas categorias principais (GUIDO S.; MULLER, 2016): problemas de classificação, quando as classes possuem valores categóricos (discretos), e problemas de regressão, quando as classes apresentam valores contínuos.

2.2.2 Aprendizagem não-supervisionada

No contexto de aprendizado não supervisionado, um dos principais pontos é que não há uma resposta ou classe "certa" ou "errada" associado aos dados de entrada. Assim, desconhecer a resposta "correta" é a melhor opção para que o aprendizado não supervisionado funcione (III, 2012). As tarefas de descrição têm como objetivo analisar e descrever um conjunto de dados sem classes pré-definidas. Essa abordagem busca identificar estruturas intrínsecas ou naturais nos dados, mesmo quando as classes não estão disponíveis. Dentre as tarefas descritivas, podemos destacar o agrupamento, a associação e a sumarização.

Este trabalho foca na tarefa de agrupamento que consiste em organizar os dados em grupos com base em sua similaridade. A ideia é que objetos pertencentes ao mesmo grupo sejam mais similares entre si do que em relação a objetos de outros grupos. Essa técnica é amplamente utilizada no aprendizado não-supervisionado para extrair informações relevantes e padrões ocultos em conjuntos de dados não rotulados.

2.3 Algoritmos de agrupamento

Algoritmos de agrupamento, também conhecidos como algoritmos de clusterização, fazem parte do aprendizado não-supervisionado, sendo usados para agrupar um conjunto de dados em grupos semelhantes, chamados de *clusters*. O objetivo é criar grupos que tenham alta semelhança interna entre os membros e baixa semelhança com o restante.Existem dois tipos clássicos de algoritmos de agrupamento: particional e hierárquico (KAUFMAN; ROUSSEEUW, 1990).

No aprendizado particional, os dados são divididos em grupos com base em atributos semelhantes, buscando a melhor divisão para que os elementos dentro de cada grupo sejam mais semelhantes entre si do que com elementos de outros grupos. Em contraste, o aprendizado hierárquico constrói uma estrutura em forma de dendrograma, com grupos formados em diferentes níveis de detalhe. Existem duas abordagens hierárquicas: aglomerativa e divisiva. Na abordagem aglomerativa, cada dado começa como um grupo individual e ocorrem fusões iterativas com base na similaridade. Na abordagem divisiva, todos os dados começam em um único grupo e são divididos em subgrupos iterativamente.

Esses dois tipos de algoritmos de agrupamento, particional e hierárquico, têm vantagens e desvantagens distintas. O particional é geralmente mais rápido e escalável, embora possa ser sensível à inicialização e ao número dos grupos. Já o método hierárquico oferece uma representação mais abrangente dos dados, mas pode ser computacionalmente mais custoso e menos escalável.

2.3.1 Algoritmos de agrupamento tradicionais

2.3.1.1 Algoritmo *K-means*

O algoritmo de agrupamento particional *K-means* é uma técnica amplamente utilizada para realizar tarefas de agrupamento. Devido à sua facilidade de implementação e abrangência de aplicações em diversas áreas, distintos pesquisadores elaboraram diferentes versões do *K-means* para melhorar a inicialização dos centroides e tornar a computação do algoritmo mais rápida (JIN; HAN, 2017).

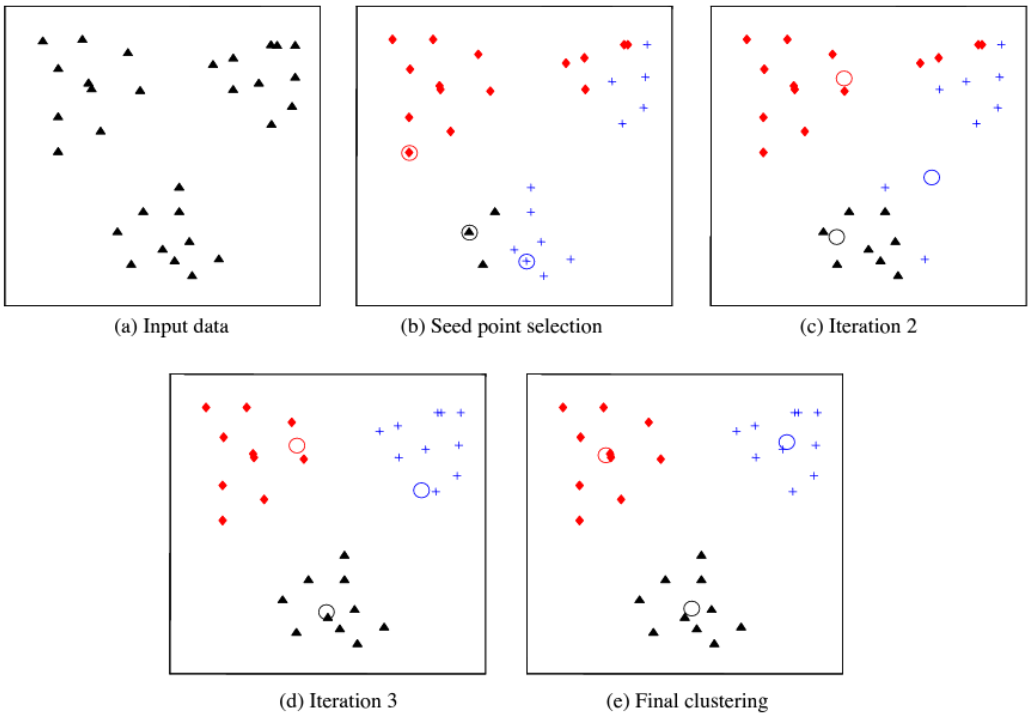
Isto tornou-se necessário em função de algumas limitações do algoritmo, tais como: o número de grupos deve ser especificado a priori, sensibilidade a *outliers*, necessidade de normalizar as variáveis caso tenham escalas muito desiguais e dificuldade com dados de alta dimensionalidade, visto que a distância entre os pontos tende a se tornar mais uniforme, dificultando a geração de agrupamentos relevantes.

O algoritmo *K-means* inicia com k grupos e atribui padrões aos grupos com o intuito de reduzir o erro ao quadrado. Assim, os principais passos do algoritmo *K-means* podem ser descritos como (JAIN; DUBES, 1988):

1. Selecione uma partição inicial com k grupos; repita as etapas 2 e 3 até os membros do grupo se estabilizarem.
2. Gere uma nova partição atribuindo cada padrão ao seu centro do grupo mais próximo.
3. Calcule novos centros dos grupos.

A Figura 2 mostra a aplicação do algoritmo de *K-means* com uma (a) entrada de duas variáveis com três grupos, (b) três pontos atribuídos inicialmente como o centro dos grupos, (c) e (d) sendo iterações intermediárias com a atualização da classe dos grupos e seus centros e (e) sendo o final do agrupamento obtido pelo algoritmo *K-means* na convergência (JAIN, 2010).

Figura 2 – Ilustração do algoritmo *K-means*.



Fonte: (JAIN, 2010).

2.3.1.2 Agglomerative Clustering

O Agrupamento Aglomerativo, *Agglomerative Clustering*, é um algoritmo hierárquico de agrupamento que inicia com cada observação sendo tratada como um grupo individual. Em seguida, ocorrem fusões iterativas dos grupos, baseadas em medidas de similaridade, até que todas as observações sejam agrupadas em um único grupo. Sendo, portanto, um método *bottom-up*, também conhecido como abordagem aglomerativa.

O algoritmo aglomerativo de agrupamento abordado, neste trabalho, é o *Ward*. Este é um método de agrupamento distintivo, fundamentado no critério clássico de soma

dos quadrados, com o intuito de formar grupos que minimizam a dispersão à medida que os grupos são combinados (MURTAGH; LEGENDRE, 2014). Além disso, este algoritmo explora o espaço euclidiano de múltiplas variáveis na busca por grupos, oferecendo uma abordagem fascinante para a análise de dados. Assim, espera-se que os agrupamentos encontrados sejam bem definidos e significativos.

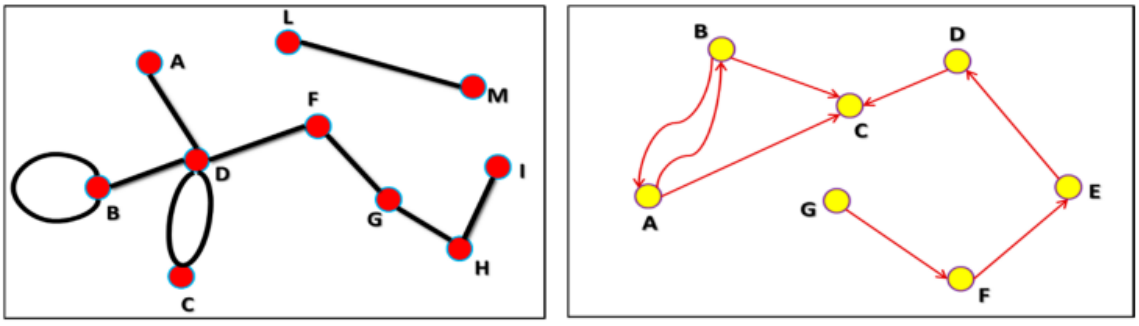
2.3.2 Algoritmos de agrupamento baseados em grafos

2.3.2.1 Definições iniciais e a importância dos grafos

É possível representar um conjunto de dados por meio de um grafo. Este é composto por um conjunto de nós e um conjunto de arestas que conectam esses nós. Os nós representam os objetos, enquanto as arestas as relações existentes entre os mesmos. Assim, é possível definir de forma formal que um grafo $G(V, E)$ representa um conjunto finito $V = v_1, v_2, \ldots v_n$, na qual os n elementos são denominados nós, juntamente com um conjunto $E = e_1, e_2, ..., e_m$ de arestas, tal que uma aresta que liga os nós v_i e $v_j \in V$ pode ser denotada por $e_{i,j}$. Além disso, dois nós são ditos adjacentes ou vizinhos se estão conectados por uma aresta.

Os algoritmos de detecção de comunidade necessitam que os dados estejam estruturados no formato de grafos. Em muitos cenários, os dados já são naturalmente relacionais, sendo que a estrutura do grafo é naturalmente estabelecida através do relacionamento entre as entidades. Por exemplo, usuários da rede social *Facebook*, naturalmente, formam um grafo relacional de amizade, no qual os nós representam os usuários e as arestas as relações de amizade. Entretanto, em alguns contextos, como dados temporais ou textuais, os dados não são naturalmente relacionais e, nesses casos, é necessário que sejam escolhidos algoritmos de construção de grafos adequados para que os dados sejam transformados. Ou seja, a escolha do método de construção dos grafos acaba sendo, muitas vezes, mais importante do que a escolha do próprio algoritmo de aprendizado em si (ZHU, 2005).

Figura 3 – Exemplo de grafos direcionados e não direcionados.



(a) Example of an undirected graph. (b) Example of the directed graph.

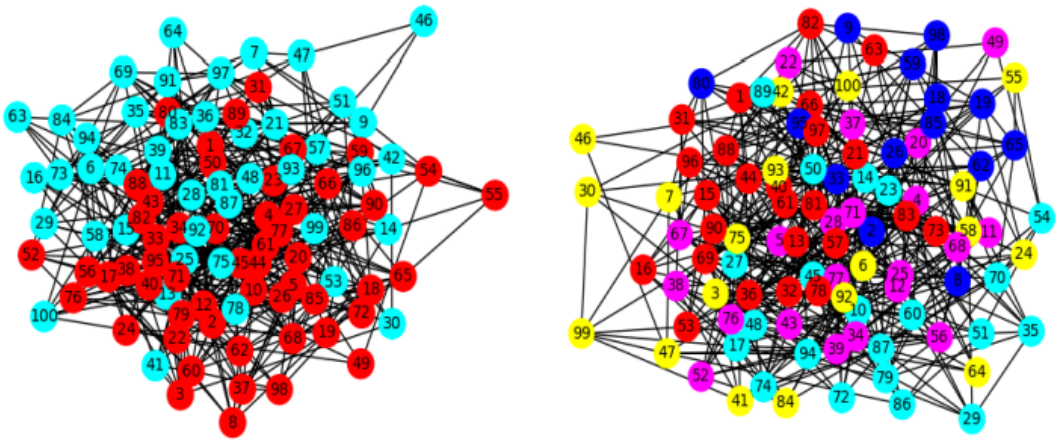
Fonte: (MAJEED; RAUF, 2020).

Alguns dos tipos mais comuns de grafos são:

- Direcionados, em que as arestas possuem direção, tendo uma orientação específica;
- Não direcionados, nos quais a conexão entre os nós é bidirecional;
- Ponderados, em que cada aresta possui um peso, influenciando no cálculo ou construção do grafo em si;
- Não ponderados, nos quais a existência ou não de nós já indica a relevância da questão, ou seja, todas as arestas possuem o mesmo peso;
- Além de outras variedades de tipos de grafos com atributos específicos que podem ser encontradas em diferentes contextos e aplicações.

Na Figura 3 há um grafo não direcionado no exemplo (a) e um direcionado no (b). Além disso, na Figura 4 tem-se um exemplo de utilização de grafos para realizar o agrupamento de uma rede social, na qual o exemplo (a) mostra um agrupamento com 2 comunidades e o (b) com 5 comunidades dados 100 usuários.

Figura 4 – Exemplo de grafos para agrupamento de rede social.



(a) Two communities in a graph **(b)** Five communities in a graph

Fonte: (MAJEED; RAUF, 2020).

2.3.2.2 *k-Nearest-Neighbours*

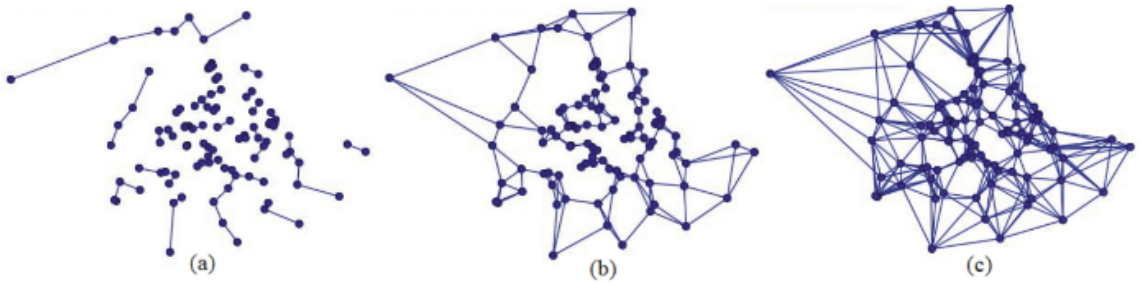
O *k-Nearest-Neighbours* (k-NN) é um método de construção de grafos não paramétrico, que é simples, porém funciona para várias situações (GUO et al., 2003).

Dado um conjunto de amostras $X = \{x_1, \ldots, x_n\}$, o k-NN considera que o vizinho mais próximo de x_i, considerando-se $i \neq j$, é x_j se este detém a menor distância a x_i. Há vários possíveis cálculos de distância, tais como Cosseno, Manhattan, ou a mais utilizada a distância Euclidiana.

O algoritmo k-NN utiliza essas distâncias para construir um grafo, na qual os k vizinhos mais próximos de cada exemplo são conectados por uma aresta. Esse grafo representa as relações de proximidade e similaridade em relação aos exemplos.

Na Figura 5 foram gerados grafos para diferentes valores de k. É possível visualizar no exemplo (a) que um valor baixo de k pode gerar um grafo desconexo. Nos exemplos (b) e (c) nota-se uma maior densidade nas arestas conforme há o incremento de k. Além disso, no canto superior de cada exemplo (b) e (c) observa-se que um nó distante é forçado a se ligar com os seus N vizinhos mais próximos, o que pode conectar exemplos bem heterogêneos.

Figura 5 – Exemplo de grafos gerados pelo k-NN dado um conjunto de dados com 100 elementos e distribuição gaussiana (a) $k = 1$,(b) $k = 3$ e (c) $k = 7$.



Fonte: (BERTON, 2016).

2.3.2.3 *Mutual k-NN*

O algoritmo *Mutual k-NN* (Mk-NN) busca diminuir a influência de pontos de dados com ruido concentrando-se no princípio dos vizinhos mútuos (DHAR et al., 2020). Ao adotar o princípio dos vizinhos mútuos, o algoritmo foca em identificar vizinhos mais próximos que possuam uma relação recíproca. Assim, esta variação é considerada mais restritiva que a sua versão original por ter como pré-requisito que seja necessário seguir o princípio dos vizinhos mútuos para o grafo ser criado. Em suma, o algoritmo k-NN é utilizado em contextos mais amplos, visto ser menos restritivo e poder identificar diversos grupos simultaneamente. Entretanto, recomenda-se a versão do Mk-NN quando busca-se identificar determinados grupos específicos ou regiões que tenham alta densidade de dados. A Figura 6 demonstra a utilização do Mk-NN para $k = 11$ em um determinado conjunto de dados, gerando o grafo mostrado.

2.3.2.4 *Sequential k-NN*

O algoritmo *Sequential k-NN* (Sk-NN) busca estabelecer novas conexões entre os nós incrementalmente, partindo de $k = 1$ até um valor definido de k_{max}. Após calcular os k vizinhos próximos para os nós e arrumá-los dado um critério de relevância, os nós do vetor ordenado são selecionados e busca-se conectá-los aos k_{max} vizinhos mais próximos

Figura 6 – Grafo construído utilizando o Mk-NN com k = 11.

Fonte: (BRITO et al., 1997).

que tenham grau inferior a k, ou seja, com menos de k conexões. Caso seja irrealizável, incrementa-se o valor de k e repete-se o processo iterativamente. Assim, o algoritmo só conclui quando todos os nós possuírem um grau maior ou igual a k_{max} dado um intervalo de k especificado pelo usuário (VEGA-OLIVEROS et al., 2014).

O objetivo principal do Sk-NN é tentar evitar a presença de *hubs*, que são nós que apresentam graus com valores muito elevados perto da média dos graus do grafo. Tais nós podem existir em grafos gerados pelos algoritmos k-NN e o Mk-NN. Com isso, o Sk-NN não gera grafos que seja perfeitamente regulares - grafos com o mesmo número de conexões, porém evitam graus muito distintos da média geral.

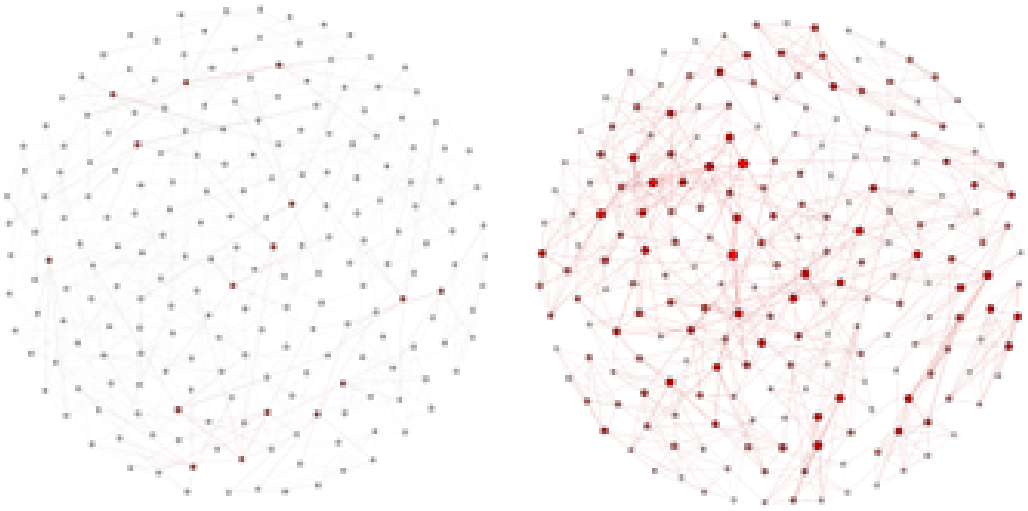
A Figura 7 ilustra uma comparação entre os grafos gerados pelos algoritmo Sk-NN e k-NN, respectivamente, com parâmetro $k = 5$ dado o mesmo conjunto de dados.

2.3.3 Algoritmos de Detecção de Comunidade

A detecção de comunidades é um problema fundamental quando trabalhamos com grafos, tendo em vista que permite revelar a existência de uma organização interna não trivial, além de ajudar a compreender as propriedades dos processos dinâmicos que ocorrem nos mesmos (YANG; ALGESHEIMER; TESSONE, 2016).

Os algoritmos de agrupamento tradicionais e os de detecção de comunidades apresentam diferenças significativas ao analisar a estrutura e os padrões em conjuntos de dados, como os grafos. Embora tenham o objetivo de identificar agregações de elementos semelhantes, há distinções em relação ao modo como operam.

Figura 7 – Grafos gerados através dos algoritmos Sk-NN e k-NN respectivamente para k=5.



Fonte: (VEGA-OLIVEROS et al., 2014).

Nos algoritmos de agrupamento tradicionais, o algoritmo recebe um conjunto de dados com seus atributos e calcula a similaridade dos mesmos através de uma medida de distância, como, por exemplo, a distância euclidiana. Para a utilização dos algoritmos de detecção de comunidades, é necessário anteriormente que o conjunto de dados passe por algoritmos que formem os grafos para depois serem aplicados os algoritmos de detecção de comunidades nos grafos já formados. Assim, os métodos de detecção de comunidade consideram além dos critérios de similaridade entre os dados, também a sua topologia.

2.3.3.1 *Fastgreedy*

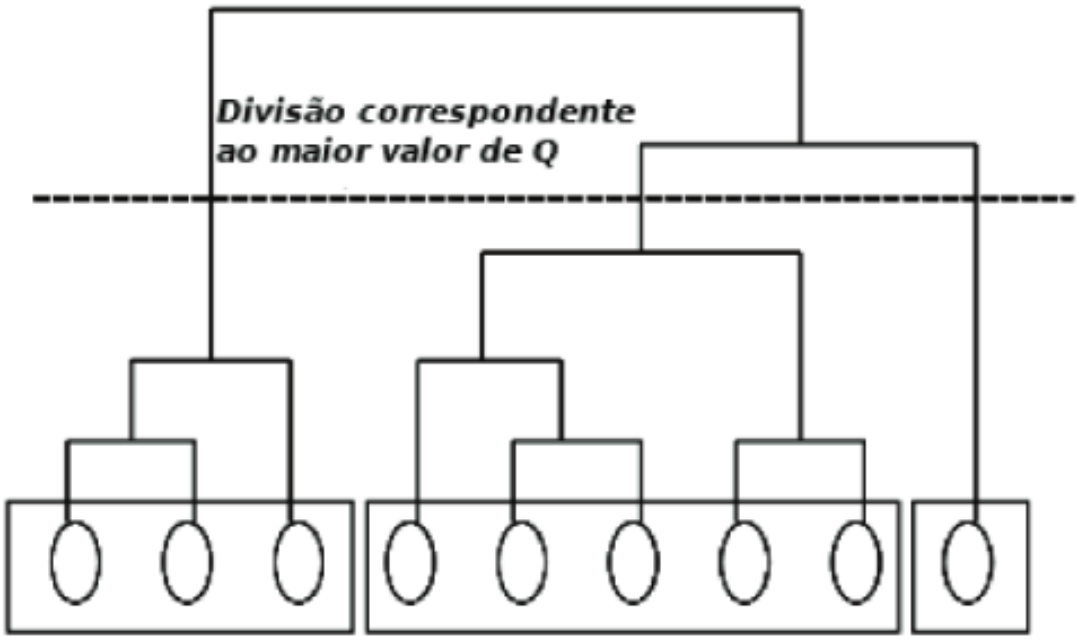
O Algoritmo *Fastgreedy* faz parte dos algoritmos hierárquicos aglomerativos, sendo fundamentado no conceito de Modularidade. Esta é uma medida que avalia a qualidade de uma divisão específica do grafo (NEWMAN; GIRVAN, 2004), analisando o grau de conectividade dos nós em comunidades distintas. Essa é uma métrica amplamente empregada em algoritmos de detecção de comunidades, com o propósito de mensurar a eficácia da subdivisão do grafo em grupos coesos. Um valor maior para a Modularidade mostra uma separação mais eficiente do grafo em comunidades (FORTUNATO, 2010). Assim, o algoritmo produz todas as possíveis divisões, calcula a Modularidade para cada uma e encontra a que tiver maior valor de Modularidade (RAGHAVAN; ALBERT; KUMARA, 2007).

Devido ao possível crescimento exponencial do número de nós, identificar todas as possíveis divisões do grafo se torna uma tarefa complexa (CLAUSET; NEWMAN; MOORE, 2004). Diante desse desafio, recorre-se ao uso de uma heurística para diminuir o

conjunto de divisões geradas. Com isso, o algoritmo utiliza uma abordagem gulosa, na qual tenta renunciar as possibilidades que não irão resultar no melhor resultado.

O algoritmo inicia atribuindo a cada nó do grafo uma comunidade individual, e em cada passo, pares de comunidades são combinados. A seleção dos pares é feita de maneira a maximizar a modularidade. O processo de combinação é repetido até que todos os nós pertençam a uma única comunidade. As combinações realizadas em cada passo são armazenadas em uma estrutura hierárquica chamada dendrograma. Após todas as iterações, o dendrograma resultante é analisado e é escolhida a divisão que apresenta o melhor desempenho em termos de modularidade. Um exemplo é a Figura 8, na qual tem-se uma representação de um dendograma com o corte gerado para o valor máximo de modularidade.

Figura 8 – Representação de um dendograma com o corte respectivo que denota o valor máximo de modularidade para a situação.



Fonte: (RAGHAVAN; ALBERT; KUMARA, 2007).

2.3.3.2 *Leading Eigenvector*

O algoritmo *Leading Eigenvector* é um método de agrupamento baseado em técnicas espectrais que utilizam propriedades algébricas de matrizes e análise de autovalores e autovetores para representar grafos (VALEJO, 2014).

Considerando uma matriz $M_{n\times n}$, $\lambda \in \mathbb{R}$ é um autovalor de M caso exista um vetor $\gamma \neq 0$ tal que $M\gamma = \lambda\gamma$. Em decorrência, γ é um autovetor associado a λ.

No contexto da teoria espectral, é comum representar um grafo através de sua matriz Laplaciana. No entanto, o algoritmo *Leading Eigenvector*(NEWMAN; GIRVAN, 2004), utiliza a matriz de modularidade B como base, na qual:

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

em que $w_{v,u}$ representa o peso da aresta que conecta os nós v e u.

No algoritmo *Leading Eigenvector*, o objetivo é analisar o autovetor γ associado ao maior autovalor positivo de B. Dado esse autovetor, é possível determinar a separação entre os nós com base nos seus sinais correspondentes. Os nós com sinais diferentes são atribuídos a comunidades distintas. Caso todos os elementos tenham o mesmo sinal, isso é um indicativo que o grafo não possui uma estrutura de comunidades bem definida (VALEJO, 2014).

2.4 Medidas de avaliação

No aprendizado não-supervisionado, avaliar os resultados é um processo desafiador, visto que não há classes ou medidas objetivas claras de desempenho para orientar a avaliação. Isso implica em confiar mais na interpretação subjetiva e na habilidade humana para validar os resultados obtidos pelos algoritmos de aprendizado não supervisionado. Como demonstrado por Hartigan (HARTIGAN, 1985), tem-se que diferentes algoritmos são corretos para diferentes propósitos, então não se pode afirmar que um é o melhor para tudo. Isto elucida a dificuldade em termos uma avaliação coerente dos resultados.

Assim, para validar um agrupamento é possível utilizar três tipos de critérios: relativos, internos e externos (JAIN; DUBES, 1988). Os índices baseados em critérios relativos comparam diversos agrupamentos para decidir qual deles é o melhor dado algum aspecto, baseando somente nos dados originais. Os que utilizam critérios internos são calculados utilizando-se exclusivamente dos atributos intrínsecas dos dados. Já o critério externo implica na comparação dos resultados com uma estrutura de referência previamente conhecida. Neste trabalho será utilizado o índice externo denominado Informação Mútua Normalizada (*Normalized Mutual Information*), uma vez que os experimentos serão realizados em dados sintéticos, ou seja, há uma partição de referência.

2.4.0.1 *Normalized Mutual Information*

A Informação Mútua Normalizada, *Normalized Mutual Information*, (NMI) é uma medida utilizada para avaliar a similaridade entre as partições geradas por um algoritmo de agrupamento em relação as classes reais dos dados, tendo sua base no conceito de informação mútua, que é uma medida estatística da dependência entre duas variáveis. Seu

objetivo é fornecer um resultado entre 0 a 1, na qual 0 indica que não há similaridade mútua entre os conjuntos e 1 indica uma correlação perfeita (STREHL; GHOSH, 2002). O NMI é particularmente útil quando sabe-se a divisão dos dados previamente ou quando se avaliam dados artificiais que apresentam uma partição de referência conhecida.

O cálculo do NMI é realizado dessa forma: Levando em consideração dois conjuntos, predito (U) e previsto (V). Calcula-se a entropia (H) para ambos:

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

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

em que $P(i)$ é a probabilidade de um objeto aleatório de U ser atribuído à classe U_i, da mesma forma para $P(j)$. Após a realização do cálculo das entropias, calcula-se também o índice de informação mútua (MI) para os conjuntos predito e previsto:

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

$P(i,j) = |U \cap V|/N$ sendo o número total de objetos. O MI pode ser reescrito 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 NMI pode ser escrito, portanto, como:

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

2.4.0.2 *Good Edges*

Good Edges foi o nome estabelecido para a medida dada pela proporção de arestas que ligam nós de categorias diferentes, referida como proporção de ϕ-arestas, apresentado por Ozaki (OZAKI et al., 2011). Assim, é possível com tal métrica analisar inicialmente o comportamento dos algoritmos de construção de grafos para os conjuntos de dados com alta dimensionalidade. Visando facilitar o entendimento, neste trabalho, foi feita a inversão da métrica original que mostra a quantidade de arestas erradas.

2.5 Dimensionalidade

Cada objeto de dado é representado e armazenado como um conjunto de atributos, como por exemplo, cor, tamanho, peso, transparência. Ao invés de utilizar o termo atributos, é possível utilizar também o termo dimensões, visto que um objeto com n atributos também pode ser representado como um ponto multidimensional em um espaço de n dimensões (VLACHOS, 2010).

2.5.1 Alta Dimensionalidade

Em AM, a alta dimensionalidade diz respeito a conjuntos de dados que possuem um número de atributos muito maior do que o número de amostras do conjunto de dados (SIRIMONGKOLKASEM; DRIKVANDI, 2019).

À medida que a dimensionalidade é ampliada, o número de variáveis ou atributos aumenta consideravelmente. Esse incremento nas dimensões resulta na esparsificação dos dados, ou seja, à medida que a dimensionalidade cresce, os dados se tornam mais dispersos e menos densos.

A premissa é que mesmo um pequeno aumento na dimensionalidade demanda um aumento muito grande no volume dos dados para manter um desempenho análogo em tarefas de AM, tais como as de agrupamento. Isso significa que, ao adicionar mais atributos aos dados, a quantidade de dados necessária para abranger todo o espaço dimensional aumenta excepcionalmente.

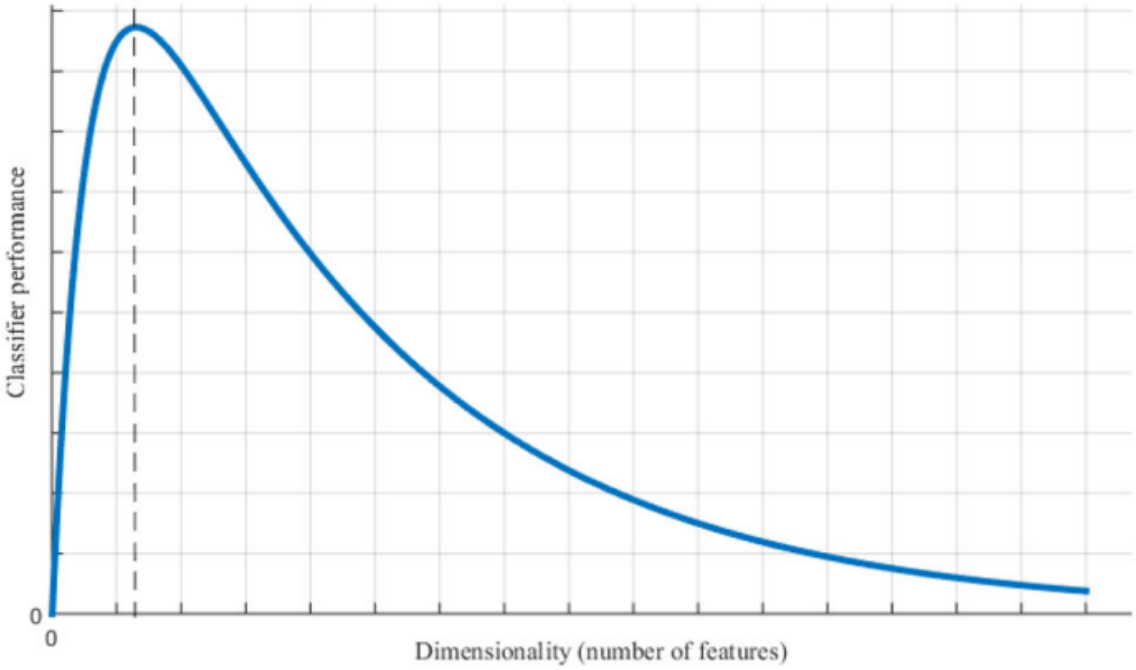
Para ilustrar essa situação, é possível realizar uma comparação utilizando o conceito de volume. À medida que novas dimensões são acrescentadas, o volume exigido para abranger o espaço dimensional se expande rapidamente, evidenciando a complexidade e os desafios na modelagem e utilização de dados com alta dimensionalidade.

2.5.2 Maldição da Dimensionalidade

A Maldição da Dimensionalidade é, em suma, um problema em que o aumento de atributos leva a um aumento expressivo da complexidade e do custo computacional. Assim, o aumento do número de atributos não implica diretamente em um aumento da performance. Pelo contrário, costuma-se haver um "ponto ótimo", no qual após o mesmo há uma queda na performance conforme mostrado na Figura 9.

Em conjuntos de dados com alta dimensionalidade, é comum encontrar informações repetitivas ou duplicadas. Para superar o problema da dimensionalidade muitos possíveis recursos para a redução da mesma foram estudados ao decorrer dos anos, os quais buscam superar os efeitos de informações redundantes ou irrelevantes, mapeando as informações realmente importantes e distintas contidas nos atributos (JIA et al., 2022).

Figura 9 – Gráfico de performance vs número de atributos para um classificador.



Fonte: (JIA et al., 2022).

3 Metodologia

Este capítulo descreve os conjuntos de dados, os métodos, algoritmos e ferramentas que foram empregados para realizar os experimentos e obter os resultados. Os algoritmos foram elaborados na linguagem de programação Python[1], com o auxílio das bibliotecas do Scikit-learn[2] e do igraph[3].

3.1 Geração dos conjuntos de dados

Para a realização deste trabalho foram utilizadas bases de dados sintéticas advindas do método *make blobs* existente na biblioteca *scikit-learn*. Tais conjuntos de dados são multiclasse e gerados alocando cada classe em, pelo menos, um grupo de pontos com distribuição normal. O método *make blobs* é utilizado para tarefas de agrupamento, visto que proporciona maior controle em relação à quantidade de centros e desvios padrões dado cada grupo.

Para a realização dos experimentos foram variados parâmetros, buscando entender melhor o comportamento dos algoritmos frente a diferentes cenários. Os parâmetros alterados para a criação dos conjuntos de dados foram:

- *n_samples*: Quantidade de amostras geradas
- *n_features*: Quantidade de atributos de cada dado
- *centers*: Quantidade de grupos gerados
- *cluster_std*: O desvio padrão dos grupos, que indica o quanto os pontos de cada grupo vão se sobrepor

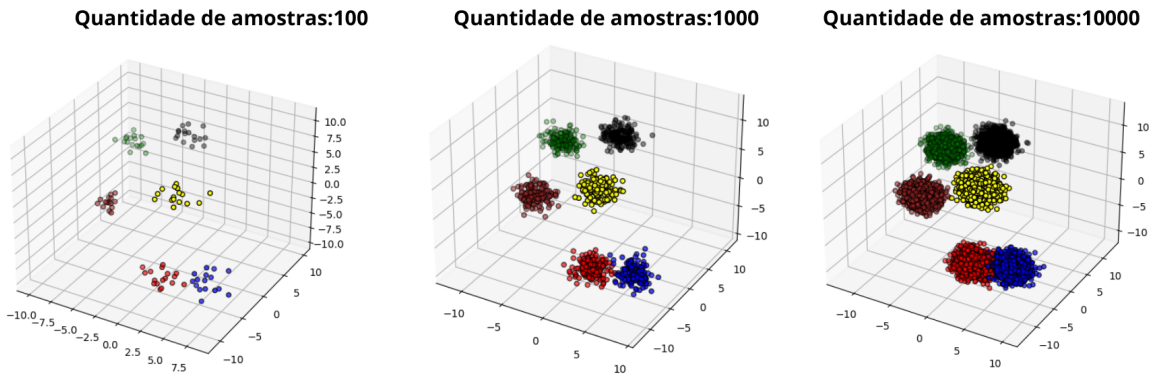
Para os demais parâmetros que não foram especificados acima, foram utilizados seus valores padrões e, portanto, não serão abordados neste trabalho.

Quando se trabalha com alta dimensionalidade um problema recorrente é em relação à visualização dos dados. Assim, buscando explicitar melhor o que a variação de cada um dos parâmetros interfere, tem-se as Figuras 10, 11, 12 para conjuntos de dados com $n\_features = 3$. Não foi realizada a diferença na visualização do número de atributos, porém as mesmas interferem na quantidade de dimensões existentes nas amostras. Assim,

[1] <https://www.python.org/>
[2] <https://scikit-learn.org/stable/>
[3] <https://python.igraph.org/en/stable/>

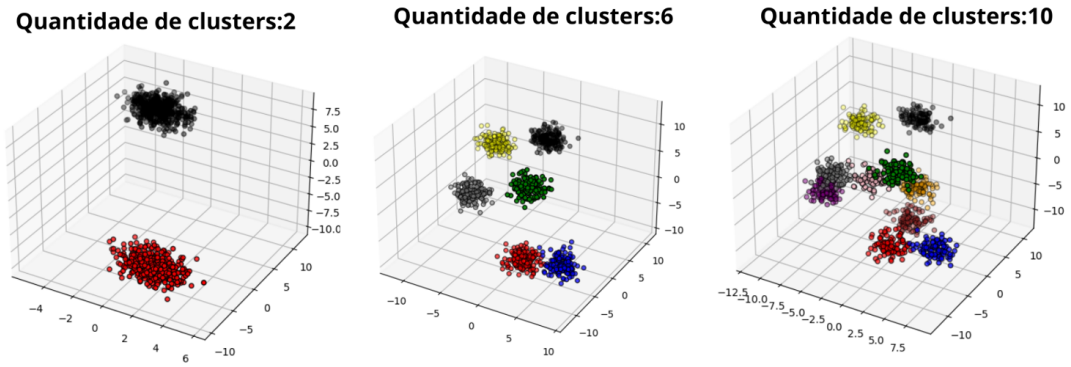
para uma dimensão poderia ter sido realizado uma visualização com um único eixo - linha reta - e com duas dimensões seriam dados em um plano.

Figura 10 – Exemplo variação da quantidade de amostras.



Fonte: Do Autor.

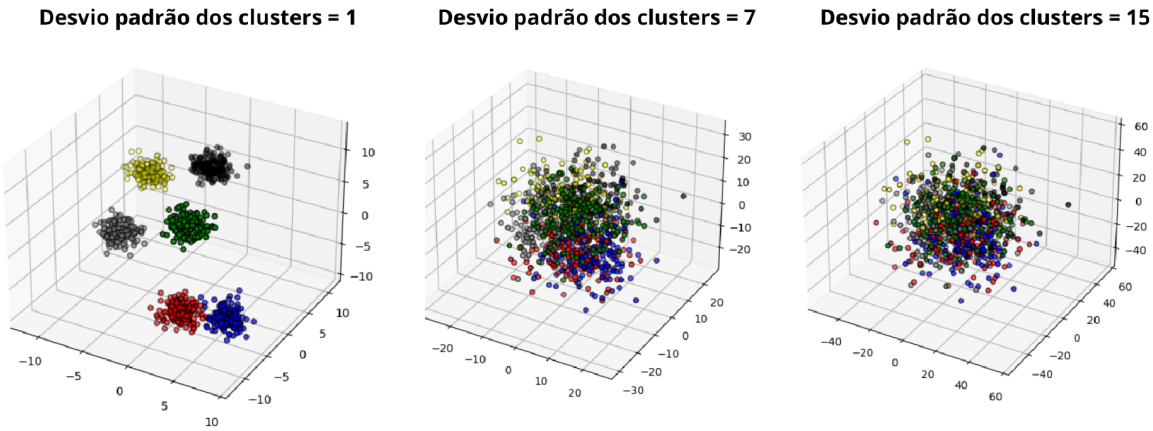
Figura 11 – Exemplo variação da quantidade de grupos.



Fonte: Do Autor.

Assim, visando avaliar o desempenho de cada algoritmo, é necessário executá-los em diferentes conjuntos de dados. Para isso, os parâmetros mencionados foram variados individualmente para os intervalos especificados na Tabela 1. Quando um parâmetro for variado, os outros valores permanecerão constantes. Entretanto, por se tratar de uma base sintética gerada quando o parâmetro escolhido for o número de classes/dimensões, além deste, será necessário variar também o desvio padrão dos grupos para simular o mundo real, na qual os dados, normalmente, acabam ficando mais esparsos em decorrência do aumento do número de classes. Para não ser realizada uma variação muito abrupta, o desvio padrão dos grupos vai seguir a proporção de ser a raiz quadrada do valor do número de classes no momento da iteração na maioria dos experimentos.

Figura 12 – Exemplo variação da sobreposição.



Fonte: Do Autor.

Tabela 1 – Intervalo de variação dos parâmetros para a geração dos conjuntos de dados.

Parâmetro	**Intervalo**	**Incremento**
Quantidade de classes	3 - 1003	5
Quantidade de amostras	100 - 1000	100
Quantidade de grupos	1 - 50	1
Sobreposição	10 - 24	1

3.2 Algoritmos Utilizados

Os algoritmos escolhidos para realizar os experimentos foram especificados no Capítulo 2, como parte da Fundamentação Teórica. A seguir estão explicitados alguns dos parâmetros fixados para a implementação dos algoritmos:

***K-means*:** Implementação do scikit-learn - class sklearn.cluster.Kmeans()

- *init*: k-means++
- *n_init*: 10
- *max_iter*: 300
- *algorithm*: Lloyd (LLOYD, 1982)

***Agglomerative Clustering*:** Implementação do scikit-learn

- *linkage*: Ward
- *compute_distances*: falso

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

- *mode*: distance

- *metric*: euclidean
- *include_self*: falso

Sk-NN Implementação própria

- *mode*: distance
- *metric*: euclidean

Fastgreedy e Leading eigenvector: Implementação do *igraph*

- *weights*: Peso da rede ponderada

3.3 Construção dos grafos

Neste trabalho focou-se em algoritmos de construção de grafos. Assim, foram escolhidos os algoritmos de construção explicados no Capitulo 3: k-NN, Mk-NN e o Sk-NN. Estes são necessários visto que os algoritmos de Detecção de Comunidade utilizam dados que estejam dispostos no formato de grafos.

3.4 Experimentos

Os experimentos foram gerados a partir da linguagem Python, empregando as bibliotecas mencionadas. A organização geral do código pode ser segmentada em três camadas de iteração:

1. Os parâmetros variáveis (quantidade de atributos, quantidade de grupos, quantidade de amostras e sobreposição dos grupos) foram selecionados um a um;

2. Para cada lista dos parâmetros variáveis, cada algoritmo foi executado dez vezes para conjuntos de dados distintos - preservando os parâmetros fixados - calculando a média e o desvio padrão do desempenho de cada algoritmo e guardando os valores;

3. Consiste em blocos de iteração para algoritmos que requerem a definição do usuário de algum parâmetro relevante para o algoritmo, na qual os algoritmos foram executados dentro de uma faixa de valores para cada um dos parâmetros e, para o cálculo da média do algoritmo para aquele conjunto de dados, foi considerado apenas o parâmetro com o qual o algoritmo apresentou melhor desempenho. Os intervalos de variação para cada parâmetro estão descritos na Tabela 2, sendo "incremento"a magnitude da variação de cada iteração no intervalo.

Tabela 2 – Intervalo de variação dos parâmetros escolhidos pelo usuário.

Algoritmo	**Parâmetro**	**Intervalo**	**Incremento**
k-NN	k	1 - 40	1
Mk-NN	k	1 - 40	1
Sk-NN	k	1 - 40	1

Assim, dado o efeito que os algoritmos de construção de grafos tem no desempenho final gerado pelos algoritmos de Detecção de Comunidades todas as combinações citadas foram realizadas, sendo escolhido o melhor valor da métrica dado os diferentes valores dos parâmetros.

3.5 Teste estatístico

Via de regra, não há um procedimento padrão para realizar uma análise comparativa do desempenho do agrupamento em conjuntos de dados diversos. Diferentes abordagens estatísticas estão disponíveis, porém a escolha de qual delas utilizar requer experiência para conseguir distinguir se as discrepâncias entre os algoritmos são genuínas ou meramente aleatórias.

Assim, foi conduzida uma avaliação estatística, de acordo as orientações sugeridas por Demšar que compreende, por exemplo, o teste ANOVA, o teste t-pareado, o teste de Friedman, entre outros, com o intuito de comparar diferentes algoritmos que operam sobre um determinado conjunto de dados (DEMŠAR, 2006). Entretanto, realizar essa avaliação não é trivial para usuários sem conhecimento aprofundado em testes estatísticos, visto que as orientações compreendem diversos métodos.

Consequentemente, visando garantir um resultado mais apurado foi utilizado o pacote *Autorank* (HERBOLD, 2020), utilizando a linguagem Python. O propósito do *Autorank* é tornar a análise estatística mais acessível para indivíduos não especializados, dado que o mesmo engloba o processo das orientações mencionadas anteriormente por meio de uma única instrução. Complementarmente, funcionalidades adicionais viabilizam a criação de representações gráficas adequadas e tabelas contendo os resultados. Para isso, é necessário que os dados sobre as populações estejam dispostos em *dataframes* da biblioteca Pandas (https://pandas.pydata.org/) do Python.

O *Autorank* emprega o seguinte procedimento para efetuar a comparação estatística: inicialmente, aplica o teste de Shapiro-Wilk para avaliar se a distribuição dos resultados provém de uma distribuição normal; a Correção de Bonferroni é empregada para aferir esses testes. Quando os dados aderem a uma distribuição normal, o Teste de Bartlett é subsequente, considerando a homogeneidade; em contrapartida, caso isso não ocorra, o Teste de Levene é utilizado. Com base nos resultados dos testes de normalidade e

homogeneidade, o *Autorank* identifica os testes e técnicas mais adequados para definir quais são os intervalos de confiança empregados para a comparação estatística.

4 Análise e discussão dos resultados

Este Capítulo tem o propósito de mostrar os resultados experimentais adquiridos com a execução dos experimentos apresentados no Capítulo 3. Assim, este Capítulo está dividido em 5 seções criadas a partir da aplicação de algoritmos em diferentes bases de dados, abordando:

1. Variação da quantidade de atributos para mensurar a qualidade dos grafos pelos algoritmos de criação de grafos;
2. Variação da quantidade de atributos;
3. Variação da quantidade de amostras;
4. Variação da quantidade de grupos;
5. Variação da sobreposição.

Dos itens 2 a 5 há a análise e discussão dos resultados em relação ao desempenho dos algoritmos tradicionais e dos algoritmos de construção de grafos associados aos de detecção de comunidade.

4.1 Variando a quantidade de atributos para mensurar a qualidade dos grafos

A Figura 13 mostra a qualidade, considerando a métrica *Good Edges*, dos grafos gerados pelos algoritmos k-NN, Mk-NN e Sk-NN com a variação da quantidade de atributos.

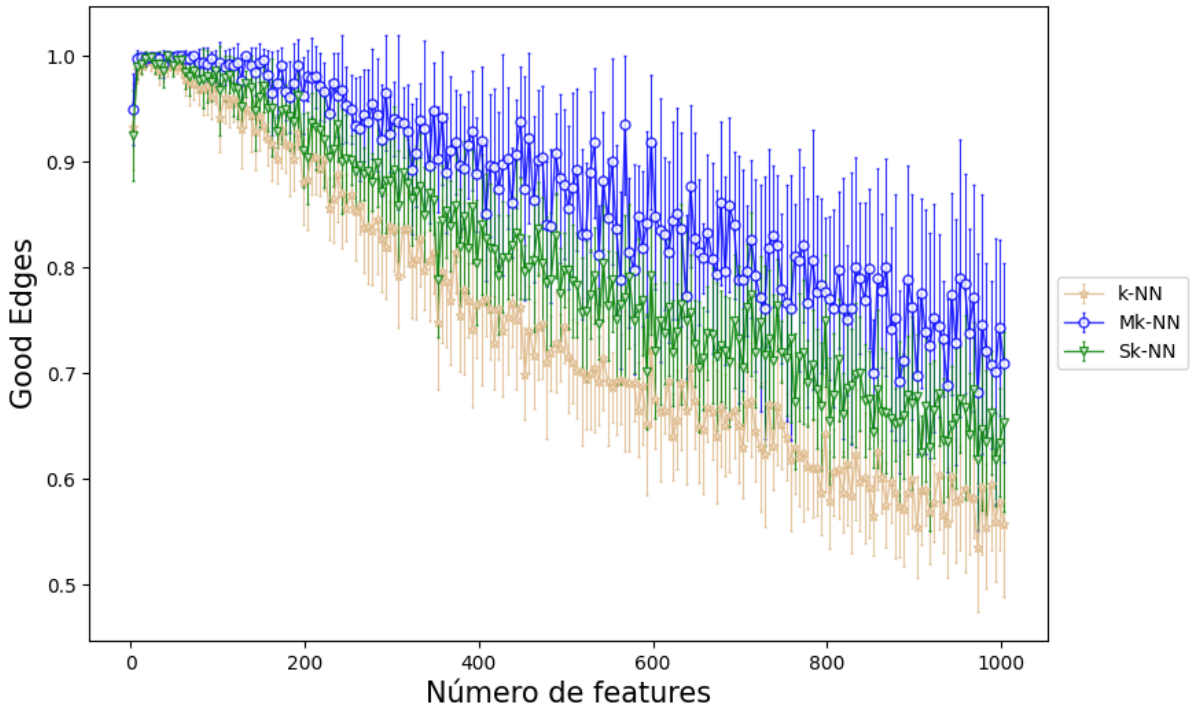
Para a criação desse experimento foram utilizados os parâmetros:

- *n_samples*: 100
- *n_features*: Variou de 3 até 1003 com incremento de 5
- *cluster_std*: Variou a raiz quadrada de 3 até 1003 com incremento de 5
- *centers*: 6

Houve a necessidade de variar tanto a quantidade de atributos quanto o desvio padrão dos grupos, tendo em vista que o *make blobs* é um método que cria uma base de dados sintética, e, buscando simular dados reais é usual que os mesmos fiquem mais

esparsos com a adição de novos atributos.A Tabela 3 foi criada com os resultados da média e desvio padrão da métrica utilizada das arestas obtidas por cada algoritmo.

Figura 13 – Qualidade dos grafos advindos dos algoritmos de construção.



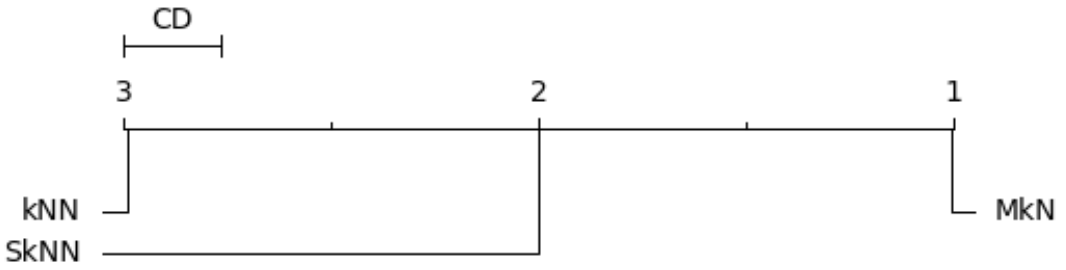
Fonte: Do Autor.

Tabela 3 – Good Edges Médio e Desvio Padrão variando o número de atributos.

Algoritmo de construção de grafos	Média	Desvio Padrão
Mk-NN	0,9207456197215903	0,07351340618925549
Sk-NN	0,8682474286484876	0,09896085304495689
k-NN	0,8225560076762887	0,12423130878122116

A análise estatística é representada pela Figura 14 dados os resultados obtidos, na qual foi conduzida para 3 populações com 201 amostras emparelhadas. O nível de significância global dos testes utilizado é alfa=0.050. O *Autorank* observou que nem todas as populações são normais. Assim, foi escolhido o teste não-paramétrico de Friedman para verificar se existem diferenças relevantes entre os valores medianos das populações. Foi utilizado o teste de Nemenyi para descobrir quais diferenças são significativas. Diferenças entre populações foram consideradas significativas apenas se a diferença da média for maior que a distância crítica CD = 0.234 do teste de Nemenyi. Em relação ao teste de Friedman, o *Autorank* rejeitou a hipótese que não há diferença na tendência central das populações. Com isso, pode-se assumir que há diferença estatisticamente significativa entre os valores medianos das populações. Para finalizar, com base no teste de Nemenyi, também foi possível assumir que todas as diferenças entre as populações são significativas. O que fica claro também pela Figura 14, na qual é possível visualizar que as distâncias entre os algoritmos de construção de grafos são maiores que a distância crítica.

Figura 14 – Comparação estatística dos resultados gerados pelo *Autorank* variando quantidade de atributos para mensurar a qualidade dos grafos.



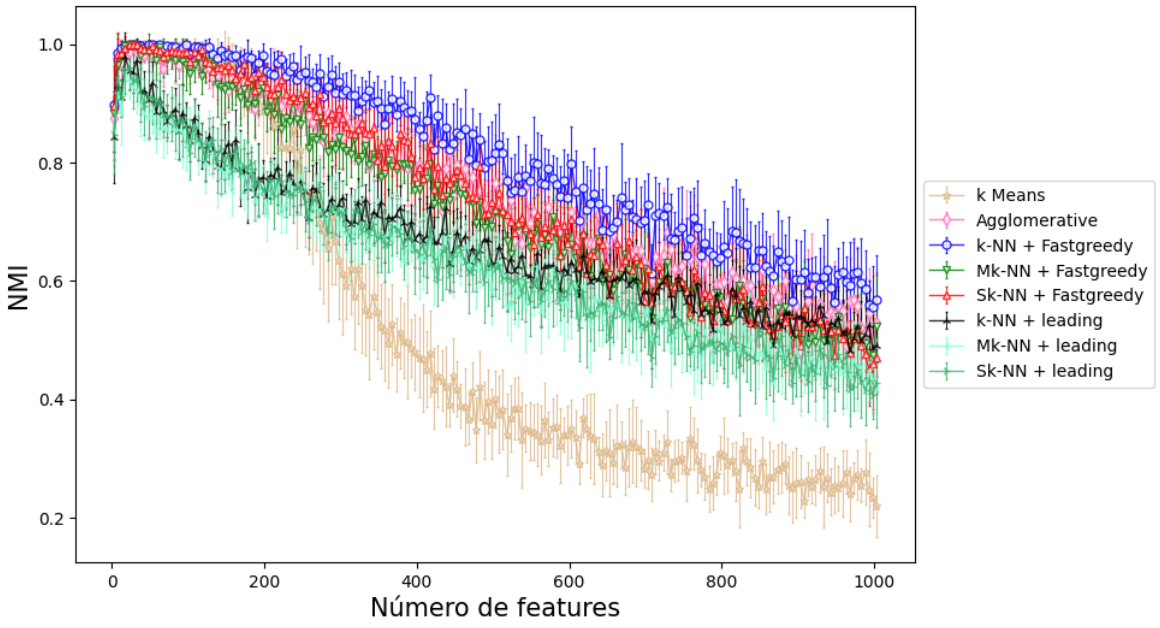
Fonte: Do Autor.

Assim, fazendo uma comparação entre a Tabela 3 e a Figura 14, nota-se que os algoritmos estão ordenados do melhor para o pior desempenho da mesma forma. A variação do Mk-NN obteve o melhor desempenho, seguido da outra variação Sk-NN, e por fim, o algoritmo k-NN.

4.2 Variando a quantidade de atributos

Foram utilizados os mesmos parâmetros fixos da seção 4.1. A Figura 15 é a demonstração do desempenho dos algoritmos tradicionais e de grafos com detecção de comunidade variando a quantidade de atributos, utilizando o NMI como métrica.A Tabela 4 foi criada com os resultados da média e desvio padrão da métrica NMI alcançados por cada algoritmo.

Figura 15 – Performance dos algoritmos variando o número de atributos.



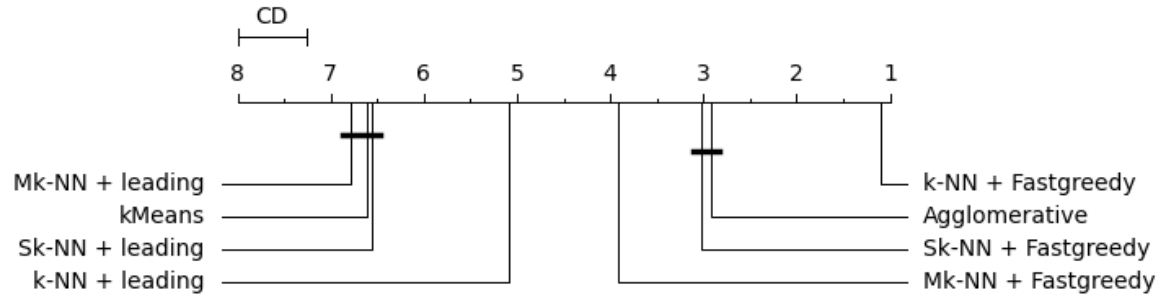
Fonte: Do Autor.

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

Algoritmos de agrupamento	Média	Desvio Padrão
k-NN + Fastgreedy	0.8845701590873611	0.1131306050229112
Sk-NN + Fastgreedy	0.8357146473907384	0.14169696045464883
Agglomerative	0.8324650421980263	0.12756857055265788
Mk-NN + Fastgreedy	0.8144820477333197	0.13943434974519792
k-NN + leading	0.7349365143851226	0.11558758864484
Sk-NN + leading	0.7144065145461567	0.13198823166100104
Mk-NN + leading	0.7062130487864544	0.12602012774118856
K-means	0.6631593988386486	0.27947757576544446

A análise estatística é representada pela Figura 16 dado os resultados obtidos, na qual foi conduzida para 8 populações com 201 amostras emparelhadas. O nível de significância global dos testes utilizado é alfa=0.050. O *Autorank* observou que nem todas as populações são normais. Assim, foi escolhido o teste não-paramétrico de Friedman para verificar se existem diferenças relevantes entre os valores medianos das populações. Foi utilizado o teste de Nemenyi para descobrir quais diferenças são significativas. Diferenças entre populações foram consideradas significativas apenas se a diferença da média for maior que a distância crítica CD = 0.741 do teste de Nemenyi. Em relação ao teste de Friedman, o *Autorank* rejeitou a hipótese que não há diferença na tendência central das populações. Com isso, pode-se assumir que há diferença estatisticamente significativa entre os valores medianos das populações. Para finalizar, com base no teste de Nemenyi, também foi possível assumir que não há diferenças significativas nos seguintes grupos: Mk-NN + *leading*, *K-means* e Sk-NN + *leading*; Sk-NN + *Fastgreedy* e *Agglomerative*. Todas as outras diferenças são significativas. O que fica claro também pela Figura 16, na qual é possível visualizar que as distâncias crítica entre os algoritmos.

Figura 16 – Comparação estatística dos resultados gerados pelo *Autorank* variando quantidade de atributos.



Fonte: Do Autor.

Assim, fazendo uma comparação entre a Tabela 4 e a Figura 16, nota-se que os algoritmos estão ordenados do melhor para o pior desempenho de forma semelhante.

É possível verificar que o algoritmo de construção de grafos utilizado impactou nos

resultados dos algoritmos de detecção de comunidades, dados as diferenças de performance existentes entre o k-NN, Mk-NN e Sk-NN, na qual a melhor performance se deu no k-NN.

Fica claro que o algoritmo de detecção de comunidade *Fastgreedy* teve resultados melhores do que o *Leading Eigenvector*. Por fim, tem-se que o k-NN + *Fastgreedy* chegou a ter resultados melhores que o algoritmo tradicional *Agglomerative*, tendo o desempenho próximo ao Sk-NN + *Fastgreedy*. Já o *K-means* teve a pior performance juntamente com o Mk-NN + *leading* e o Sk-NN + *leading*.

Com esse experimento da variação de atributos foi possível perceber a interferência da quantidade dos mesmos e consequente efeito da Maldição da Dimensionalidade 2.5.2. Além disso, tem-se que os resultados isolados dos algoritmos de construção de grafos apresentados na Seção 4.1 se mostraram diferentes dos resultados no contexto de agrupamento, visto que o Mk-NN obteve os melhores resultados na construção dos grafos, porém o pior entre os algoritmos de construção de grafos quando associados aos algoritmos de detecção de comunidade para tarefas de agrupamento. Assim, há um indício que grafos mais regulares parecem ter uma melhor performance no contexto de agrupamento, mesmo conectando nós de classes distintas. Enquanto o Mk-NN tem a possibilidade de gerar muitos *hubs*, o que pode impactar negativamente nos algoritmos de detecção de comunidades.

4.3 Variação da quantidade de amostras

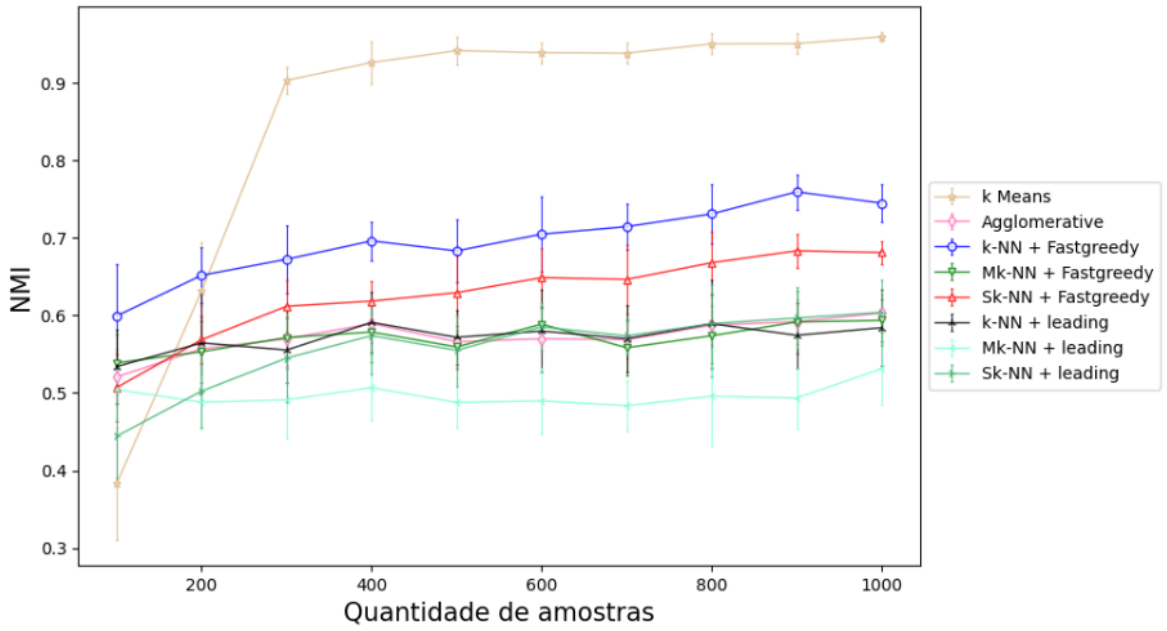
Para a criação desse experimento foram utilizados os parâmetros:

- *n_samples*: Varia de 100 até 1000 com incremento de 100
- *n_features*: 200
- *cluster_std*: 20
- *centers*: 6

A Figura 17 é a demonstração do desempenho dos algoritmos tradicionais e de grafos com detecção de comunidade variando a quantidade de amostras.A Tabela 5 foi criada com os resultados da média e desvio padrão da métrica NMI alcançados por cada algoritmo dado a variação da quantidade de amostras.

A análise estatística é representada pela Figura 18 dado os resultados obtidos, na qual foi conduzida para 8 populações com 10 amostras emparelhadas. O nível de significância global dos testes utilizado é alfa = 0.050. O *Autorank* observou que nem todas as populações são normais. Assim, foi escolhido o teste não-paramétrico de Friedman para verificar se existem diferenças relevantes entre os valores medianos das populações. Foi utilizado o teste de Nemenyi para descobrir quais diferenças são significativas. Diferenças

Figura 17 – Performance dos algoritmos variando a quantidade de amostras.



Fonte: Do Autor.

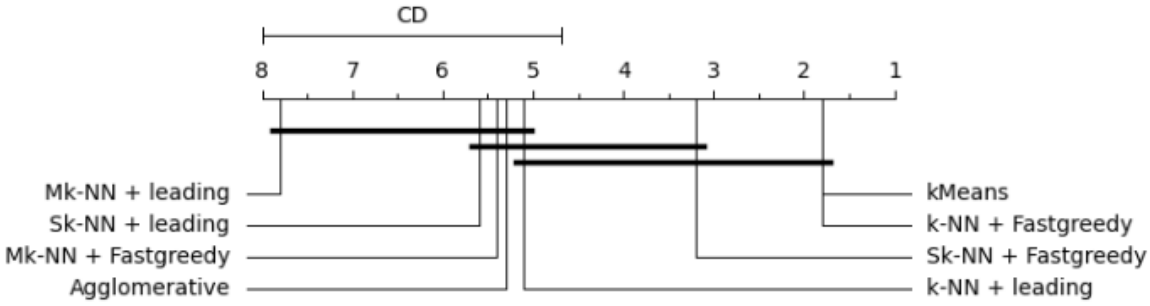
Tabela 5 – NMI Médio e Desvio Padrão variando a quantidade de amostras.

Algoritmo de agrupamento	**Média**	**Desvio Padrão**
K-means	0.7822629962132792	0.2176423544173613
k-NN + Fastgreedy	0.6735458170774843	0.043656274239751824
Sk-NN + Fastgreedy	0.6011767598650359	0.05374162313994231
k-NN + leading	0.5656029302407495	0.01851343853196444
Mk-NN + Fastgreedy	0.5639318162912205	0.01670519621746292
Agglomerative	0.5631083778526407	0.022967683668714173
Sk-NN + leading	0.5355225199621401	0.05142897867737157
Mk-NN + leading	0.4950172307887857	0.009237119226853097

entre populações foram consideradas significativas apenas se a diferença da média for maior que a distância crítica CD = 3.320 do teste de Nemenyi. Em relação ao teste de Friedman, o *Autorank* rejeitou a hipótese que não há diferença na tendência central das populações. Com isso, pode-se assumir que há diferença estatisticamente significativa entre os valores medianos das populações. Para finalizar, com base no teste de Nemenyi, também foi possível assumir que não há diferenças significativas nos seguintes grupos:Mk-NN + *leading*, Sk-NN + *leading*, Mk-NN + *Fastgreedy*, *Agglomerative* e k-NN + *leading*; Sk-NN + *leading*, Mk-NN + *Fastgreedy*, *Agglomerative*, k-NN + *leading* e Sk-NN + *Fastgreedy*; k-NN + *leading*, Sk-NN + *Fastgreedy*, *K-means* e k-NN + *Fastgreedy*. Todas as outras diferenças são significativas.

É possível verificar novamente que o algoritmo de construção de grafos utilizado impactou, de certa forma, nos resultados dos algoritmos de detecção de comunidades, dado que o k-NN + *leading* apareceu entre os melhores algoritmos em desempenho na

Figura 18 – Comparação estatística dos resultados gerados pelo *Autorank* variando a quantidade de amostras.



Fonte: Do Autor.

frente do próprio *Agglomerative* e do Mk-NN + *Fastgreedy*. Assim, tem-se que com a variação do número de amostras o algoritmo de construção de grafos k-NN teve um melhor desempenho.

Diferentemente dos demais experimentos, o algoritmo *K-means* obteve performance superior ao *Agglomerative*. Além disso, neste experimento tivemos uma interferência menor do que nos demais em relação ao algoritmo de detecção de comunidades.

4.4 Variação da quantidade de grupos

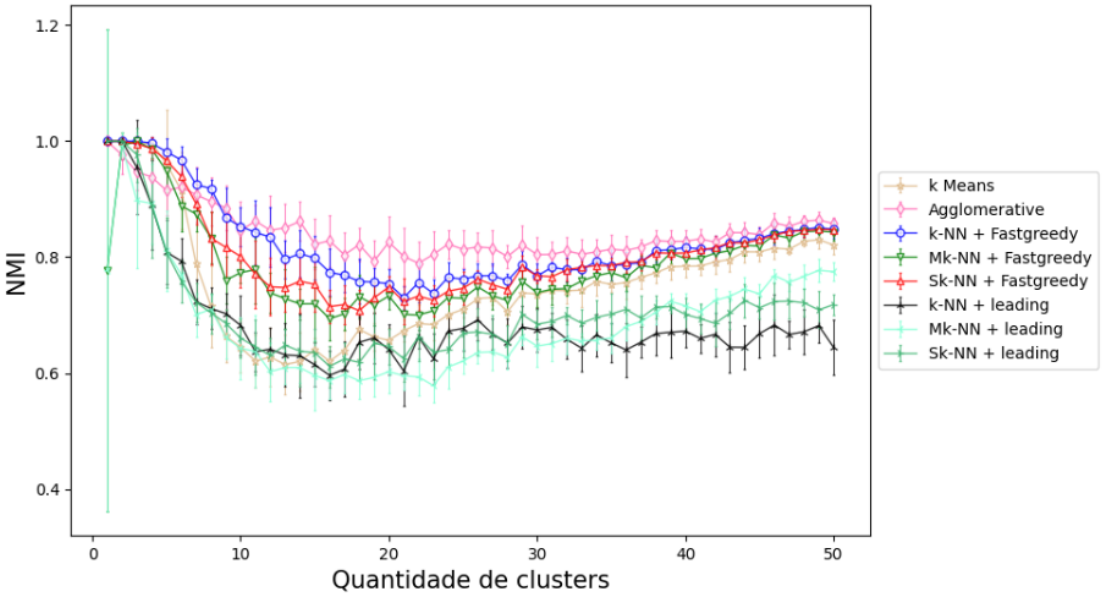
Para a criação desse experimento foram utilizados os parâmetros:

- *n_samples*: 100
- *n_features*: 200
- *cluster_std*: Raiz quadrada de 200
- *centers*: Variou de 1 até 50 com incremento de 1

A Figura 19 é a demonstração do desempenho dos algoritmos tradicionais e de grafos com detecção de comunidade variando a quantidade de grupos, utilizando o NMI como métrica.A Tabela 6 foi criada com os resultados da média e desvio padrão da métrica NMI alcançados por cada algoritmo dada a variação da quantidade de grupos.

A análise estatística é representada pela Figura 20 dados os resultados obtidos, a qual foi conduzida para 8 populações com 50 amostras emparelhadas. O nível de significância global dos testes utilizado é alfa = 0.050. O *Autorank* observou que nem todas as populações são normais. Assim, foi escolhido o teste não-paramétrico de Friedman para verificar se existem diferenças relevantes entre os valores medianos das populações. Foi utilizado o teste de Nemenyi para descobrir quais diferenças são significativas. Diferenças entre populações foram consideradas significativas apenas se a diferença da média for

Figura 19 – Performance dos algoritmos variando a quantidade de grupo.



Fonte: Do Autor.

Tabela 6 – NMI Médio e Desvio Padrão variando a quantidade de grupos.

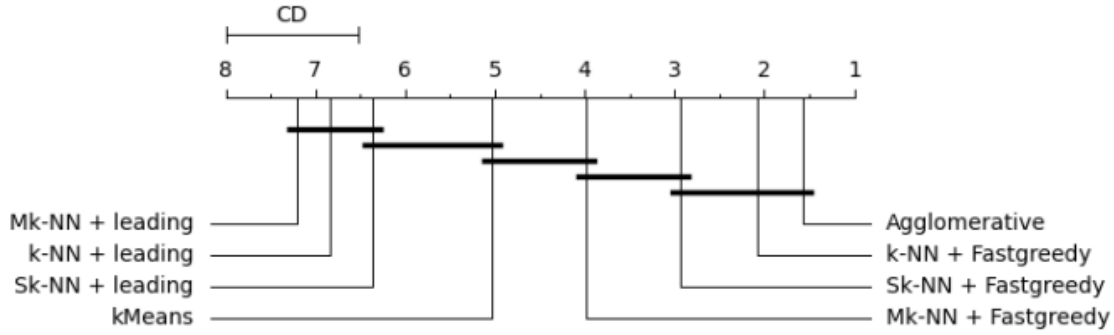
Algoritmos de agrupamento	**Média**	**Desvio Padrão**
Agglomerative	0.8593107297749705	0.05797745628769266
k-NN + Fastgreedy	0.8444708127161785	0.09076521313454697
Sk-NN + Fastgreedy	0.8180267441385197	0.09671005205953591
Mk-NN + Fastgreedy	0.792771293498851	0.09237285709532168
K-means	0.7580568608783506	0.12998866309044774
k-NN + leading	0.7137067520601768	0.11636374260171453
Sk-NN + leading	0.7197410616692879	0.11450800152277843
Mk-NN + leading	0.6877657529821505	0.1065795742014029

maior que a distância crítica CD = 1.485 do teste de Nemenyi. Em relação ao teste de Friedman, o *Autorank* rejeitou a hipótese que não há diferença na tendência central das populações. Com isso, pode-se assumir que há diferença estatisticamente significativa entre os valores medianos das populações. Para finalizar, com base no teste de Nemenyi, também foi possível assumir que não há diferenças significativas nos seguintes grupos: Mk-NN + *leading*, k-NN + *leading* e Sk-NN + *leading*; Sk-NN + *leading* e *K-means*; *K-means* e Mk-NN + *Fastgreedy*; Mk-NN + *Fastgreedy* e Sk-NN + *Fastgreedy*; Sk-NN + *Fastgreedy*, k-NN + *Fastgreedy* e *Agglomerative*. Todas as outras diferenças são significativas.

Assim, fazendo uma comparação entre a Tabela 6 e a Figura 20, nota-se que os algoritmos estão ordenados do melhor para o pior desempenho de forma semelhante.

É possível verificar que o algoritmo de construção de grafos utilizado não foi tão impactante nesse experimento, entretanto o k-NN e o Sk-NN foram melhores em geral que o Mk-NN.

Figura 20 – Comparação estatística dos resultados gerados pelo *Autorank* variando a quantidade de grupos.



Fonte: Do Autor.

Além disso, fica claro que o algoritmo de detecção de comunidade *Fastgreedy* teve resultados melhores do que o *Leading Eigenvector*. Ficou claro, tendo em vista que todos os algoritmos que o utilizam tiveram desempenho superior. Por fim, tem-se que o *Agglomerative* teve performance melhor, tendo o desempenho próximo ao Sk-NN + *Fastgreedy* e o k-NN + *Fastgreedy*.

4.5 Variação da sobreposição

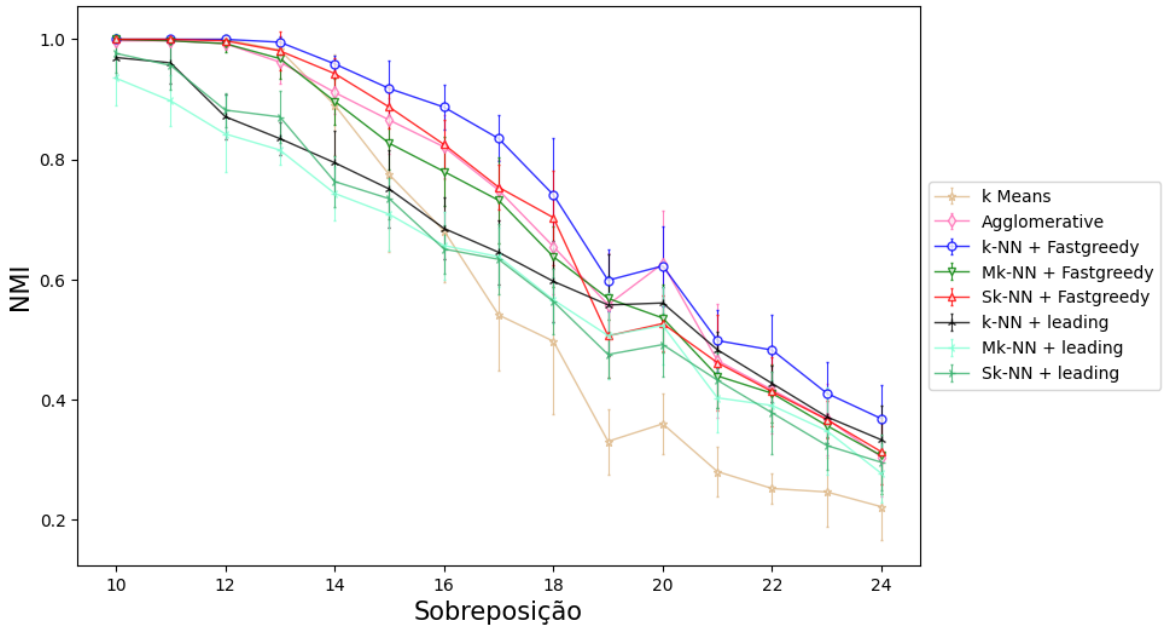
Para a criação desse experimento foi utilizado os parâmetros:

- *n_samples*: 100
- *n_features*: 200
- *cluster_std*: Variou de 10 até 24 com incremento de 1
- *centers*: 6

A Figura 21 é a demonstração do desempenho dos algoritmos tradicionais e de grafos com detecção de comunidade variando a sobreposição (desvio padrão dos grupos), utilizando o NMI como métrica. A Tabela 7 foi criada com os resultados da média e desvio padrão da métrica NMI alcançados por cada algoritmo dada a variação da sobreposição.

A análise estatística é representada pela Figura 22 dados os resultados obtidos, a qual foi conduzida para 8 populações com 15 amostras emparelhadas. O nível de significância global dos testes utilizado é alfa = 0.050. O *Autorank* observou que todas as populações são normais. Assim, foi escolhido o teste de Bartlett para homogeneidade. Com isso, foi testada a hipótese nula e, no fim, assumiu-se que os dados são homocedásticos. Foi utilizado o teste ANOVA de medidas repetidas para descobrir se as diferenças são significativas dado os valores médios das populações. O teste ANOVA de medidas repetidas

Figura 21 – Performance dos algoritmos variando a sobreposição.



Fonte: Do Autor.

Tabela 7 – NMI Médio e Desvio Padrão variando a sobreposição.

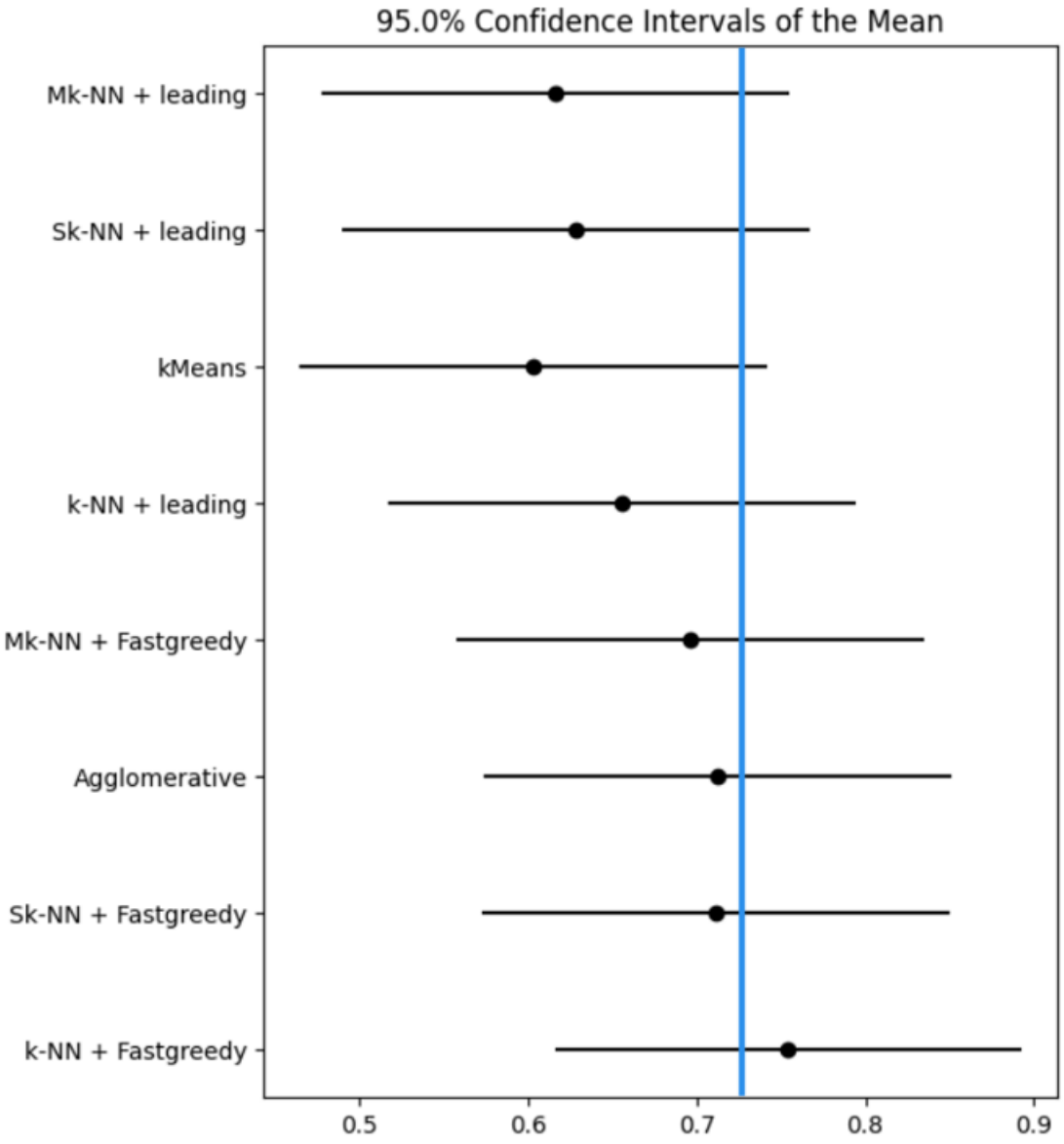
Algoritmos de agrupamento	**Média**	**Desvio Padrão**
k-NN + Fastgreedy	0.8737893351951451	0.17231818086484674
Sk-NN + Fastgreedy	0.8429558062868151	0.19669249893848065
Agglomerative	0.838768745527186	0.18526347823669329
Mk-NN + Fastgreedy	0.8254734649356024	0.1938943583075388
K-means	0.7635641357956339	0.2669384780615633
k-NN + leading	0.7626925767000031	0.1677913230532409
Sk-NN + leading	0.7473275979114432	0.1913792910247433
Mk-NN + leading	0.7248690158138129	0.16771225879253926

permitiu que se assumisse que há uma diferença significativa. Para finalizar, com base no teste de Tukey HSD, assumiu-se que não há diferenças significativas entre os grupos.

Para identificar quais algoritmos possuem semelhança estatística, é suficiente desenhar uma linha vertical. Todos os algoritmos que são cruzados por essa linha pertencem a um conjunto que não apresenta diferença estatisticamente significativa. Isso é exemplificado pela linha azul na Figura 22.

Com isso, por mais que a Tabela 7 e a Figura 22 demonstrem comportamento semelhante em relação a ordenação do melhor para o pior desempenho, tem-se que não há diferença estatística significativa para se ter uma conclusão mais aprofundada.

Figura 22 – Comparação estatística dos resultados gerados pelo *Autorank* variando a sobreposição.



Fonte: Do Autor.

5 Conclusão

Este trabalho compara os algoritmos de construção de grafos e Detecção de Comunidades, utilizando os algoritmos de agrupamento tradicionais como baseline, em conjuntos de dados sintéticos e que apresentam alta dimensionalidade. É necessário destacar que a escolha de valores ideais para os diversos parâmetros dos algoritmos interfere no resultado final apresentado. Com isso, podem ocorrer distorções no desempenho observado em detrimento da escolha arbitrária dos parâmetros sem um ajuste fino dos melhores parâmetros de cada algoritmo para cada situação.

A partir dos experimentos, foi possível constatar que os algoritmos possuem diferença estatisticamente significativa em relação à medida de desempenho adotada. Além disso, nota-se uma divergência entre a métrica do *Good Edges* e o futuro desempenho dos algoritmos nos demais experimentos para a métrica NMI, visto que a ordem parece invertida. Isto se dá pelo fato do Mk-NN ter apresentado os melhores resultados em termos de arestas classificadas corretamente, porém em todos os experimentos subsequentes obteve menor desempenho que o k-NN e o Sk-NN independente do algoritmo de Detecção de comunidades.

Na maioria dos casos, o algoritmo de Detecção de Comunidade *Fastgreedy* obteve melhores resultados independentemente do algoritmo de construção de grafos adotado. Entretanto, em geral, o algoritmo de construção de grafos k-NN obteve o melhor desempenho ao longo dos experimentos frente às demais variações. Isto tornou-se ainda mais claro no experimento de variação das amostras 4.3, na qual o k-NN + *leading* obteve resultado superior ao Mk-NN + *Fastgreedy*.

Outro ponto é que os algoritmos de grafos associados aos de criação de grafos obtiveram resultados semelhantes ou até mesmo melhores que os algoritmos de agrupamento tradicionais utilizados como baseline mostrando-se viáveis ao uso.

Foi possível constatar também que a sobreposição é o parâmetro que mais interfere nos resultados finais do NMI, visto que dados esparsos em alta dimensionalidade representam um problema complicado para os algoritmos lidarem.

Assim, dado todos os experimentos realizados, é possível concluir que os algoritmos de construção de grafos associados aos de detecção de comunidade são uma opção factível para dados em alta dimensionalidade. Além disso, a escolha dos algoritmos de construção de grafos interferem diretamente no desempenho final bem como a escolha dos algoritmos de detecção de comunidade.

5.1 Trabalhos Futuros

Para prosseguir com este trabalho, foram identificadas algumas opções que poderiam contribuir para a melhoria do estudo, trazendo mais *insights* e clareza para o tema:

- Fazer uso de bases de dados reais em que se possa observar um comportamento mais próximo com a curva da Maldição da Dimensionalidade representada na Figura 9, tendo em vista que é difícil simular a esparsividade existente no mundo real;
- Fazer uso da biblioteca *GridSearch* do scikit-learn com o intuito de otimizar os parâmetros escolhidos para a execução dos algoritmos de construção de grafos;
- Utilizar diferentes algoritmos de construção de grafos e de detecção de comunidades comparados a algoritmos tradicionais de agrupamento, porém, algoritmos mais novos e com mais potência computacional;
- Empregar o método de Análise de Componentes Principais (PCA) para a realização de análises nas bases de dados com alta dimensionalidade, buscando verificar se a performance quando o método é empregado se equipara a dos conjuntos com baixa dimensionalidade.

Referências

BERTON, L. *Construção de redes baseadas em vizinhança para o aprendizado semissupervisionado*. Tese (Doutorado) — Universidade de São Paulo, 2016. Citado na página 29.

BRITO, M. et al. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics Probability Letters*, v. 35, n. 1, p. 33–42, 1997. ISSN 0167-7152. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167715296002131>. Citado na página 30.

CLAUSET, A.; NEWMAN, M. E. J.; MOORE, C. Finding community structure in very large networks. *Phys. Rev. E*, American Physical Society, v. 70, p. 066111, Dec 2004. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevE.70.066111>. Citado na página 31.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, v. 7, p. 1–30, 2006. Citado na página 41.

DHAR, J. et al. A weighted mutual k-nearest neighbour for classification mining. *CoRR*, abs/2005.08640, 2020. Disponível em: <https://arxiv.org/abs/2005.08640>. Citado na página 29.

ERTEL, W. *Introduction to Artificial Intelligence*. [S.l.]: Springer, 2018. Citado na página 23.

FACELI, K. et al. *Inteligência Artificial: Uma abordagem de Aprendizagem de Máquina*. [S.l.]: LTC, 2011. Citado na página 24.

FORTUNATO, S. Community detection in graphs. *Physics Reports*, Elsevier BV, v. 486, n. 3-5, p. 75–174, feb 2010. Disponível em: <https://doi.org/10.1016%2Fj.physrep.2009.11.002>. Citado na página 31.

GUIDO S.; MULLER, A. C. *Introduction to Machine Learning with Python: a guide for data scientists*. [S.l.]: O'reilly, 2016. Citado na página 24.

GUO, G. et al. Knn model-based approach in classification. In: SPRINGER. *OTM Confederated International Conferences"On the Move to Meaningful Internet Systems"*. [S.l.], 2003. p. 986–996. Citado na página 28.

HARTIGAN, J. A. Statistical theory in clustering. *Journal of Classification*, v. 2, n. 1, p. 63–76, 12 1985. ISSN 1432-1343. Disponível em: <https://doi.org/10.1007/BF01908064>. Citado na página 33.

HERBOLD, S. Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, The Open Journal, v. 5, n. 48, p. 2173, 2020. Disponível em: <https://doi.org/10.21105/joss.02173>. Citado na página 41.

III, H. D. *A Course in Machine Learning*. [S.l.]: ciml, 2012. Citado na página 24.

JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, v. 31, n. 8, p. 651–666, 2010. ISSN 0167-8655. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR). Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865509002323>. Citado na página 26.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988. Disponível em: <http://portal.acm.org/citation.cfm?id=46712>. Citado 2 vezes nas páginas 25 e 33.

JIA, W. et al. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, v. 8, n. 3, p. 2663–2693, 06 2022. ISSN 2198-6053. Disponível em: <https://doi.org/10.1007/s40747-021-00637-x>. Citado 2 vezes nas páginas 35 e 36.

JIN, X.; HAN, J. *K-Means Clustering*. Boston, MA: Springer US, 2017. 695–697 p. ISBN 978-1-4899-7687-1. Disponível em: <https://doi.org/10.1007/978-1-4899-7687-1_431>. Citado na página 25.

KAUFMAN, L.; ROUSSEEUW, P. *Finding Groups in Data: An Introduction To Cluster Analysis*. [S.l.: s.n.], 1990. ISBN 0-471-87876-6. Citado na página 25.

LLOYD, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, v. 28, n. 2, p. 129–137, 1982. Citado na página 39.

MAJEED, A.; RAUF, I. Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. *Inventions*, v. 5, n. 1, 2020. ISSN 2411-5134. Disponível em: <https://www.mdpi.com/2411-5134/5/1/10>. Citado 2 vezes nas páginas 27 e 28.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill Education(ISE Editions), 1997. Citado 2 vezes nas páginas 19 e 23.

MURTAGH, F.; LEGENDRE, P. Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? *Journal of Classification*, v. 31, n. 3, p. 274–295, October 2014. ISSN 1432-1343. Disponível em: <https://doi.org/10.1007/s00357-014-9161-z>. Citado na página 27.

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 31 e 33.

OZAKI, K. et al. Using the mutual k-nearest neighbor graphs for semi-supervised classification on natural language data. In: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Portland, Oregon, USA: Association for Computational Linguistics, 2011. p. 154–162. Disponível em: <https://aclanthology.org/W11-0318>. Citado na página 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 31 e 32.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Pearson, 2016. Citado na página 23.

SIRIMONGKOLKASEM, T.; DRIKVANDI, R. On regularisation methods for analysis of high dimensional data. *Annals of Data Science*, v. 6, n. 4, p. 737–763, 12 2019. ISSN 2198-5812. Disponível em: <https://doi.org/10.1007/s40745-019-00209-4>. Citado 2 vezes nas páginas 20 e 35.

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 34.

VALEJO, A. D. B. *Refinamento multinível em redes complexas baseado em similaridade de vizinhança*. Tese (Doutorado) — Universidade de São Paulo, 2014. Citado 2 vezes nas páginas 32 e 33.

VEGA-OLIVEROS, D. A. et al. Regular graph construction for semi-supervised learning. In: IOP PUBLISHING. *Journal of physics: Conference series*. [S.l.], 2014. v. 490, n. 1, p. 012022. Citado 2 vezes nas páginas 30 e 31.

VLACHOS, M. Dimensionality reduction. In: _____. *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. p. 274–279. ISBN 978-0-387-30164-8. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_216>. Citado na página 35.

YANG, Z.; ALGESHEIMER, R.; TESSONE, C. J. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, v. 6, n. 1, p. 30750, 2016. ISSN 2045-2322. Disponível em: <https://doi.org/10.1038/srep30750>. Citado na página 30.

YINGFAN, L.; HONG, C.; JIANGTAO, C. *Revisiting k-Nearest Neighbor Graph Construction on High-Dimensional Data : Experiments and Analyses*. 2021. Citado na página 20.

ZHU, X. *Semi-supervised learning literature survey*. [S.l.], 2005. Citado na página 27.