

Comparativo teórico-prático dos algoritmos de clusterização K-Means e DBSCAN

Vinicius C. Rocha¹, Wellington C. ¹

¹Universidade Federal de Uberlândia (UFU)
Uberlândia – MG – Brazil

Resumo. *Este trabalho possui como objetivo o comparativo teórico-prático de duas técnicas de clusterização, Kmeans e DBSCAN. Para realizar o comparativo teórico neste artigo é estudado a teoria de algumas fontes na internet e os pseudocódigos das duas técnicas. Para o comparativo prático foi realizado experimentos práticos contemplando as etapas de análise de base de dados pré-processamento, mineração considerando variadas bases de dados e pós-processamento. Os resultados práticos de comparativos encontrados contemplam em sua maioria os resultados teóricos esperados.*

1. Introdução

Nas últimas décadas o uso e armazenamento de dados acompanhou o ritmo crescente da expansão do uso da tecnologia global surgindo o termo Big Data que se refere à quantidade massiva de dados armazenados. Com isso, oportunidades de analisar estes conjuntos para finalidades específicas surgiram e uma das principais técnicas para este fim é a **clusterização** que se ramifica em diversas técnicas bastante estudadas na modernidade.

Este trabalho aborda o conceito de clusterização se especificando nas técnicas de agrupamento **K-Means** e **DBSCAN** visando comparar ambas do ponto de vista **teórico-prático** ao contrastar os resultados para diferentes bancos de dados disponibilizados em plataformas online ou gerados.

Os algoritmos K-means e DBSCAN são técnicas amplamente estudadas e utilizadas na mineração de grupos e são aplicados em uma variedade de domínios, incluindo marketing, biomedicina, e análise geoespacial [Nvidia 2023].

1.1. Motivações

Considerando o contexto da técnica de agrupamento, quando comparado este a outras técnicas, clustering é mais independente do domínio da pesquisa que é utilizado devido a não utilizar rótulos o que é de extrema importância para cenários onde é necessário a análise de bases de dados sem uma classificação específica.

Além disso, é uma técnica com diversas aplicações como a mineração de texto (extração de informações por meio da identificação de padrões e tendências de escritas), segmentação da base (segmentação de indivíduos com base em seus padrões), extração de conhecimento da web (com base em htmls), entre outras [Bijuraj 2013].

Considerando o escopo de técnicas escolhidas, K-Means é uma técnica popular, bastante simples de se implementar e bastante rápida quando comparada as demais técnicas por não precisar calcular distancias entre pares de pontos (apenas de pontos aos pontos centrais definidos) [Ellis 2022b].

Já o DBSCAN é interessante por ser robusto em cenários com base de dados distribuídas em formatos irregulares, ser bastante insensível a outliers, por não necessitar da definição de quantidade de clusters e ser relativamente rápido [Ellis 2022a].

1.2. Fundamentos

Por este projeto ter sido realizado utilizando as técnicas de clusterização K-Means e DBSCAN, nesta seção será abordado a fundamentação teórica dos conceitos de clusterização, das técnicas utilizadas e o comparativo esperado entre ambas pela literatura.

1.2.1. Clusterização

Clustering ou agrupamento é uma técnica de análise de dados usada em aprendizado de máquina para identificar grupos de objetos semelhantes em conjuntos de dados. Essa abordagem faz parte do **aprendizado não supervisionado**, na qual busca encontrar padrões nos dados sem a necessidade de rótulos pré-existentes, em contraste com os métodos supervisionados que devem possuir rótulos pré-definidos. Técnicas de agrupamento também podem ser utilizadas para detectar anomalias, ou seja, pontos de dados que não se encaixam em nenhum grupo, nos capítulos seguintes é possível a visualização destes.

O conceito de semelhança entre objetos da base empregado para o agrupamento é dependente do cenário e técnica aplicada e normalmente se utiliza **métricas de distância** entre objetos como a **distância euclidiana**, **distância de manhattan**, etc. Com base na métrica as técnicas decidem se o dado será agrupado ou não em um grupo específico.

1.2.2. K-Means

K-Means ou K-Médias é uma técnica presente no escopo de clustering que realiza o agrupamento de uma base de dados se baseando em uma **medida de similaridade** de objetos a grupos específicos definidos por pontos que podem ou não pertencer à base de dados denominados **centróides**. No modelo é definido uma versão inicial do hiperparâmetro de centróides que serve para iniciar o processo iterativo de agrupamento, e por utilizar o mesmo, o modelo é dependente do hiperparametro. A métrica de similaridade normalmente utilizada para o cálculo é a menor distância euclidiana.

Considerando os centroides iniciais a aplicação do modelo iterativamente realiza o agrupamento de objetos ao centroide mais próximo atual seguido pelo ajuste dos centroides com base nas médias dos valores das dimensões dos objetos agrupados a este. Com isso os valores dos centroides mudam a cada iteração, quando não mudar o algoritmo converge para uma solução que pode na maioria das vezes não ser ótima.

Pela técnica ser dependente dos centroides iniciais, mudar o hiperparametro pode resultar em agrupamentos diferentes além do parâmetro, servir como o indicativo do número de clusters que serão gerados ao fim do processo de agrupamento.

```

Input:
     $D = \{t_1, t_2, \dots, t_n\}$  // Set of elements
     $K$  // Number of desired clusters
Output:
     $K$  // Set of clusters
K-Means algorithm:
    Assign initial values for  $m_1, m_2, \dots, m_k$ 
    repeat
        assign each item  $t_i$  to the clusters which has the closest mean;
        calculate new mean for each cluster;
    until convergence criteria is met;

```

Figure 1. Pseudocódigo K-Means [Mohd et al. 2012]

1.2.3. DBSCAN

O DBSCAN (agrupamento espacial de aplicações com ruído baseado em densidade) é o segundo algoritmo analisado. Enquanto o K-Means utiliza o princípio de centroides, o DBSCAN tem o conceito de agrupamento baseado em densidade. Em resumo, essa classe de algoritmos de agrupamento se preocupa mais com a densidade de pontos no espaço do que com a forma que um grupo pode ter. Na prática, o algoritmo recebe os hiper-parâmetros denominados Eps e MinPts, que representam, respectivamente, um raio ao redor de um ponto e uma quantidade mínima de pontos nesta vizinhança. Com esses parâmetros, é possível tomar a principal métrica do DBSCAN: a densidade das regiões de cada pontos. Todos os pontos que tiverem MinPts ou mais em sua Eps-vizinhança, são ditos de estarem em regiões de alta densidade. Assim, falando de forma superficial, o agrupamento é feito de forma com que todos os pontos em uma mesma região de alta densidade acabem no mesmo grupo.

Esse agrupamento por regiões densas é feito com o que geralmente é chamado de expansão de grupo. Escolhe-se um ponto que esteja em uma região de alta densidade e cria-se um novo grupo a partir dele. O processo de expansão é fazer com que todos os seus Eps-vizinhos também façam parte de seu grupo e então continuar a expansão a partir de todos os novos pontos que também são considerados de alta densidade. Os pontos que forem alcançados pela expansão mas estiverem em regiões de baixa densidade, são adicionados ao grupo mas não são usados na expansão. Estes pontos são chamados de pontos de borda.

A expansão continua até que não haja mais para onde expandir, e então tenta-se criar outros grupos a partir de outros pontos ainda não agrupados. Quando não houver mais pontos desagrupados, todos vão estar fazendo parte de um grupo ou rotulados como ruído.

1.2.4. Comparativo K-Means e DBSCAN

Por ser um algoritmo baseado em etapas incrementais de cálculos simples o K-Means é considerado um algoritmo simples de se entender e aplicar além de funcionar em velocidades interessantes considerando base de dados maiores quando se comparado com o DBSCAN. Em contrapartida, a técnica é dependente e variável com relação ao hiper-parâmetro de centroides iniciais fornecido, o que pode resultar em resultados diferentes

em execuções distintas. Outro ponto negativo é a forma que o algoritmo realiza o agrupamento, conforme considerado apenas a distância mínima para o agrupamento existe a tendência de se agrupar em **formatos elípticos** não considerando a topologia e densidade dos objetos além de ser **suscetível a ruídos** por estes se distanciarem das médias formadas por terem distâncias maiores [Sharma 2020].

Já o DBSCAN, por se **basear na densidade** dos pontos da base, funciona bem em cenários onde existem muitos outliers, pode ser utilizado para a detecção destes e forma agrupamentos de forma variável considerando a forma definida pela densidade dos pontos. Por outro lado, existe uma maior complexidade de entendimento para a implementação do mesmo, principalmente em cenários onde é aplicado mudanças no algoritmo para eficiência, além disso, o algoritmo não funciona bem em cenários onde os dados são esparsos devido a este ser baseado em densidade. Assim como o K-Means, o DBSCAN também é dependente de hiperparâmetros, em seu escopo é dependente do número de pontos dentro do raio e o tamanho do raio para definir o agrupamento baseado em densidade, o qual gera resultados diferentes com parâmetros diferentes [Sharma 2020].

Devido às diferenças, o uso de cada uma das ferramentas deve se basear no cenário de aplicação e não somente nas técnicas, deve-se considerar o contexto e aspectos da base de dados a ser analisada.

Métrica	K-Means	DBSCAN
Complexidade de implementação	Simple	Complexa
Dependente de Hiperparâmetros	Sim (Centróides iniciais)	Sim (Eps e MinPts)
Número de clusters deve ser definido	Sim (Depende do número de centróides)	Não
Agrupar em formas arbitrárias	Não (Formato elíptico)	Sim (Baseado na densidade)
Funciona bem com outliers	Não	Sim (Usado como detector)
Funciona bem com bases esparsas	Sim (Independente da densidade)	Não
Eficiente em tempo	Sim (Melhor que o DBSCAN)	Não

Figure 2. Tabela com resumo do comparativo K-Means X DBSCAN

2. Desenvolvimento

O projeto e experimentos foram desenvolvidos na linguagem **Rust** definidos em 5 principais partes sendo elas a escolha dos datasets de comparação, **pré-processamento** envolvendo a tratativa da entrada de dados do banco, a **mineração de dados** envolvendo o processo de clusterização pelas duas técnicas estudadas, o **pós-processamento** incluindo a verificação de métricas dos dois modelos com visualização de gráficos particionados em 2 dimensões de cada base e o **comparativo** entre os resultados obtidos. Neste capítulo serão abordadas cada uma das 5 etapas citadas anteriormente além dos motivos do uso da linguagem Rust.

2.1. Uso da linguagem Rust

A linguagem Rust está em crescente adoção da comunidade de programadores que acreditam que a mesma substituirá em grande parte a linguagem C para aplicações que exigem alta eficiência de processamento devido a esta permitir o controle de baixo nível, ser próxima em termos de eficiência da linguagem C++ e possuir funcionalidades que C e C++ não possuem para facilitar a rotina dos programadores de baixo nível [Ernerfeldt 2022].

Seu uso para o projeto é devido a esta ser considerada eficiente quando se com-
parada às linguagens mais utilizadas para realizar a implementação dos algoritmos de clustering como, por exemplo, Python e JavaScript(Node.js), além de ser estruturada para evitar problemas de memória e permitir o controle de baixo nível nas estruturas de dados o que pode ser importante para aplicações de Machine Learning.

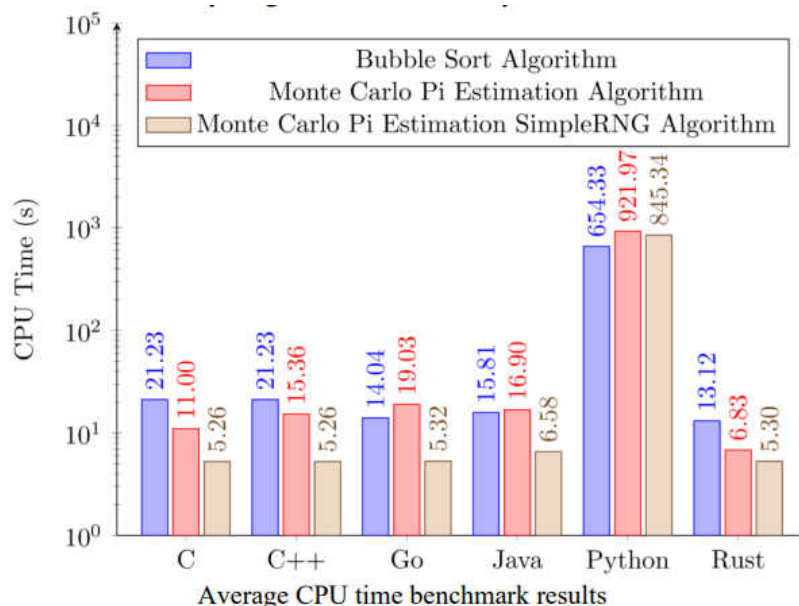


Figure 3. Benchmark CPU Time results [Fairhead 2022]

2.2. Bases de dados escolhidas

Para o estudo e comparação das duas técnicas de clustering foram selecionadas 3 bases disponíveis na plataforma **Kaggle** e outra criada com um formato específico na finalidade de testar o agrupamento realizado entre os algoritmos. Abaixo os gráficos de duas dimensões de cada uma das bases escolhidas.

Cada base de dados foi escolhida com uma finalidade específica de validação aos pontos de comparação apresentados no comparativo do capítulo anterior sendo: A base iris por ser utilizada por padrão por vários testes pela comunidade e por ter formato elíptico em suas classes com densidade equilibrada, as bases gmm e de arcos por possuírem topologia com densidade adequada e formato não elíptico para verificar se o DBSCAN funciona melhor quando comparado ao K-Means e por fim a Mall Costumers por possuir dados esparsos para verificar se o K-Means possui vantagem de agrupamento quando comparado ao DBSCAN.

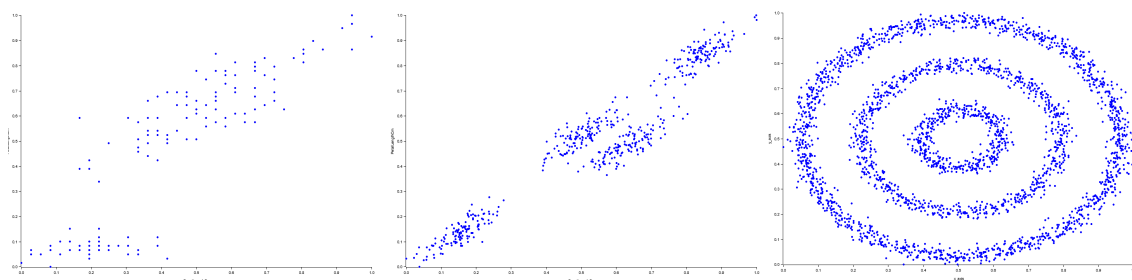


Figure 4. Bases de dados Iris, Clustering gmm disponíveis no kaggle e Base autogerada de arcos circulares

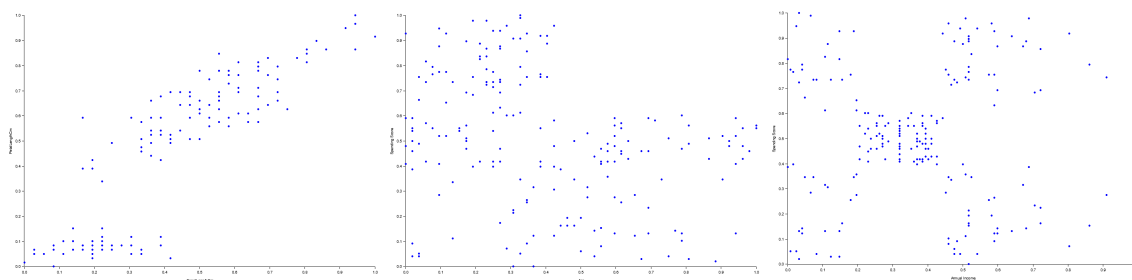


Figure 5. Base de dados Mall Costumers disponível no kaggle (Combinações de 3 dimensões)

2.3. Pré-processamento dos dados

Todas as bases de dados utilizadas foram recebidas no formato CSV com campos no formato numérico, não foram utilizados dados categóricos por não ser a finalidade deste artigo o foco no pré-processamento. A etapa se baseou nas principais partes:

- **Leitura da base de dados**
- **Conversão em um tipo abstrato de dados(TAD):** Com a finalidade de dar eficiência aos algoritmos de mineração
- **Normalização dos dados:** No intervalo $[0, 1]$

A normalização de dados utilizada seguiu a fórmula: $\frac{Xi - \min(X)}{\max(X) - \min(X)}$, onde X é um atributo da base e Xi o atributo relativo ao objeto em específico.

2.4. Mineração de dados

Esta etapa realiza a extração de grupos com base no TAD obtido pelo passo de pré-processamento utilizando as duas técnicas de clustering, K-Means e DBSCAN, para ambos os casos o mesmo TAD é enviado para manter o mesmo cenário entre ambas as técnicas e, portanto imparcialidade para obtenção de experimentos mais concisos.

2.4.1. K-Means

O algoritmo criado é baseado no pseudo-código descrito no capítulo de fundamentação teórica deste artigo. O algoritmo utiliza uma estrutura de índices para evitar mudanças na base de dados o que comprometeria a posterior execução do DBSCAN. As principais partes do algoritmo é dada abaixo:

- **Definição do grupo de centroides inicial:** É passado da função principal de execução(main) ao algoritmo alguns pontos aleatórios da base que definem os centroides e a quantidade de clusters resultantes.
- **Início do algoritmo:** É reservado recursos para realizar as iterações do algoritmo e o TAD do grupo de clusters principal.
- **Iterações - Criação de cluster temporário e calculo de distâncias:** Ao início um grupo de cluster temporário é criado e utilizado para o anexo de objetos próximos ao invés do grupo de clusters principal, o motivo é a comparação posterior de objetos para verificar se o processo convergiu.
- **Iterações - Comparação ordenada entre os grupos de clusters:** Como o TAD utilizado para abstrair os grupos de clusters é ordenado e possui índices, a comparação entre dois clusters é eficiente, basta comparar se os índices são os mesmos ordenadamente, por isso, a comparação é $O(n)$ considerando a quantidade de objetos da base.
- **Iterações - Atribuição de novos centroides:** No caso dos dois grupos de clusters serem diferentes o grupo principal de clusters recebe o temporário e é realizado o cálculo de médias dos pontos de cada grupo para formar os novos clusters e é iniciado uma nova iteração.
- **Iterações - Finalização:** No caso dos dois grupos serem iguais o modelo entra em convergência e o processo é então finalizado.

Ao final do algoritmo é retornado o TAD que possui todos os grupos criados e em convergência como resultado ao dataset juntamente com os centroides iniciais enviados.

2.4.2. DBSCAN

Nossa implementação do DBSCAN foi fortemente baseada na descrição original em [Martin Ester 1996]. As principais partes do código são dadas, de forma simplificada, abaixo:

- Primeiramente, o fluxo principal é a aplicação da expansão para todos os pontos que estão desagrupados no momento. Como mostra a figura 6.




```

for i in 0..dataset.0.len() {
    match dataset.0[i].cluster_label {
        ClusterLabel::Undefined => {
            let should_increment = expand_cluster(dataset, i, current_cluster_id, eps, min);
            if should_increment {
                current_cluster_id += 1;
            }
        }
        _ => {}
    }
}

```

Figure 6. Fluxo de nível mais alto do DBSCAN

- A função de expansão tem início com a criação de uma lista de expansão que é inicializada com os vizinhos do nó central. Caso este nó esteja em uma região de baixa densidade, ele recebe o rótulo de ruído e a função termina, já que não se pode construir um novo grupo a partir dali. Caso contrário, o nó recebe o rótulo de grupo atual e o código continua para a parte principal da expansão. A figura 7 mostra esta seção.



```
let mut expansion = neighbourhood(dataset, index, eps);
if expansion.len() < min {
    dataset.0[index].cluster_label = ClusterLabel::Noise;
    return false;
}

dataset.0[index].cluster_label = ClusterLabel::ClusterId(cluster_id);
.
.
.
```

Figure 7. Início da função de expansão

- Sabendo que o ponto central da expansão está em uma região de alta densidade, esta parte do código realiza a expansão propriamente dita. Enquanto a lista de expansão não estiver vazia, seu primeiro elemento é removido e analisado. Caso ele já tenha um rótulo de grupo, nada é feito, pois ele já foi analisado antes. Caso o rótulo seja de ruído, ele apenas recebe o rótulo de grupo atual, sendo assim descoberto como um nó de borda. Por último, no caso de não ter rótulo algum ainda, ele recebe o rótulo de grupo atual e sua vizinhança é calculada. Se, e apenas se, sua vizinhança o tornar parte de uma região de alta densidade, seus vizinhos são adicionados à lista de expansão. A figura 8 demonstra o que foi descrito.

Por último, é interessante notar a função de descoberta de vizinhança, que é mostrada na figura 9. Nossa implementação contém o algoritmo mais trivial para tal tarefa: verificar a distância de todos os pontos. Esta abordagem foi escolhida por sua simplicidade, mas é importante notar que essa busca completa tem complexidade temporal linear, o que faz o algoritmo ter complexidade quadrática.

Porém, como é argumentado no próprio artigo original ([Martin Ester 1996]), esse problema de busca pode ser resolvido em tempo logarítmico, se os pontos forem armazenados com as estruturas de dados adequadas. Sendo, assim, possível ter o DBSCAN com tempo $O(n \log(n))$.

2.5. Pós-processamento dos modelos

Considerando os modelos retornados da etapa de mineração(TAD), na etapa foram realizados a extração de algumas métricas e visualizações sendo elas:


```

while expansion.len() != 0 {
  let current = expansion.pop_front().unwrap();
  match dataset.0[current].cluster_label {
    ClusterLabel::Undefined => {
      dataset.0[current].cluster_label = ClusterLabel::ClusterId(cluster_id);
      let mut current_expansion = neighbourhood(dataset, current, eps);
      if current_expansion.len() >= min {
        expansion.append(&mut current_expansion);
      }
    }
    ClusterLabel::Noise => {
      dataset.0[current].cluster_label = ClusterLabel::ClusterId(cluster_id);
    }
    ClusterLabel::ClusterId(_) => {}
  }
}
true

```

Figure 8. Fim da função de expansão

- **Coeficiente de silhueta global:** Inclui o coeficiente para objetos, clusters e global - Para a extração da métrica é enviado o TAD juntamente com a base de dados, o TAD descreve os grupos contendo índices para a base de dados.
- **Visualização de planos:** A etapa recebe o TAD de modelo a base de dados e as dimensões da base que irá realizar a plotagem, como o TAD possui a diferenciação entre pontos de centro borda e ruído é possível fazer a plotagem considerando essas diferenças em cores, para isso, é escolhido circularmente 5 combinações diferentes de cores para pontos de centro e borda (Azul, amarelo, verde, ciano e lilás) e a cor vermelho para outliers, no caso do modelo possuir mais que 5 grupos as cores se repetem.
- **Cálculo de tempo para os algoritmos de mineração:** Outra métrica extraída foi o tempo gasto considerando o comparativo entre os dois modelos, entretanto a métrica foi extraída fora do escopo do código-fonte (No terminal).

Com as métricas geradas e visualizações é possível verificar os resultados dos experimentos.

2.6. Resultados dos experimentos

Para manter o cenário similar para as duas técnicas foi utilizado o mesmo dataset e definido métricas de forma experimental para manter o mesmo número de clusters entre os agrupamentos da mesma base de dados. A seguir são exibidos as visualizações dos planos para cada uma das bases de dados seguida pela tabela de métricas considerando tempo e coeficiente de silhueta.

Com base nos agrupamentos e resultados obtidos é possível observar para a base:

- **Íris:** Como foi mencionado na seção de base de dados sobre a base íris, a base é um dataset organizada com partições em formatos esféricos com ótima densidade

```
(0..dataset.0.len()).filter(|index2| {
  let dist = euclidean_distance(dataset, index, *index2);
  dist <= eps && dist > 0.0
}).collect()
```

Figure 9. Função de descoberta de vizinhança de complexidade de tempo linear

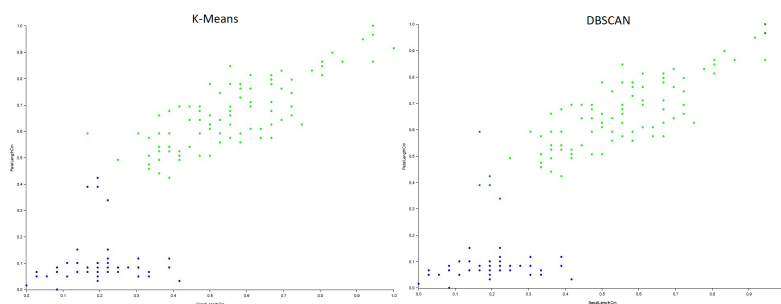


Figure 10. Base de dados Íris

sendo esperado que ambos os algoritmos agrupassem de forma semelhante, embora os coeficientes de silhueta dos agrupamentos não foram iguais o coeficiente global é bem próximo com erros na casa de 4 casas decimais.

- **Clustering GMM:** Pela teoria era esperado que o DBSCAN fosse melhor no dataset por este conter grupos separados com densidades interessantes o que foi o resultado obtido pelo experimento. Enquanto o K-Means realizou partições dividindo regiões de alta densidade o DBSCAN soube particionar evitando a quebra de classes com espaçamentos semelhantes. Devido ao formato ser relativamente esférico os coeficientes de silhuetas em ambos os cenários foram bons com destaque do DBSCAN.
- **Mall Costumers:** Este é esperado um retorno pior pelo DBSCAN devido ao espalhamento dos dados da base assim como é obtido no experimento, enquanto o DBSCAN se perde no agrupamento devido à falta de regiões de alta densidade possuindo um menor coeficiente o K-Means consegue particionar grupos de forma mais eficiente possuindo um melhor coeficiente de silhueta. É possível inferir algumas observações sobre os grupos selecionados pelos modelos ao observar os agrupamentos, por exemplo: O grupo azul se destina a objetos com menor idade e estes geralmente possuem menor renda anual e tendem a ficar mais tempo no shopping.
- **Circular:** Nesta base era esperado a diferença notória da limitação do K-Means que o DBSCAN resolve, o agrupamento elíptico. Embora visualmente é perceptível que o DBSCAN agrupa de forma correta o coeficiente de silhueta fica

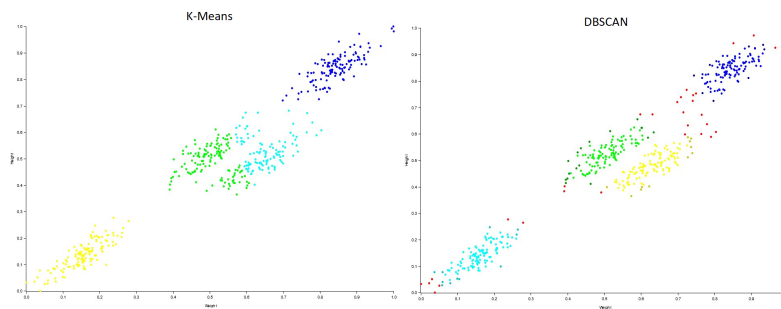


Figure 11. Base de dados Clustering GMM

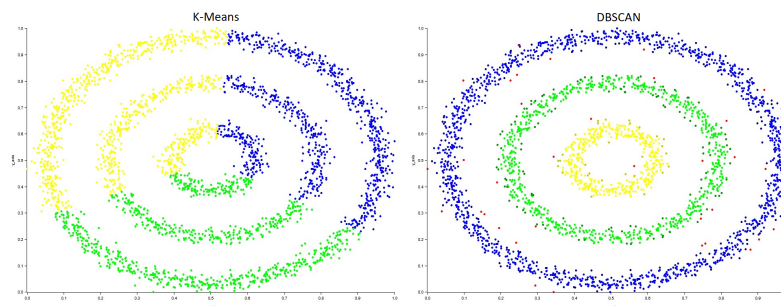


Figure 12. Base de dados Circular

negativo para esta técnica, isso se deve ao formato dos dados se distanciarem muito um dos outros por não serem grupos com espalhamento elíptico. Outro destaque para essa base é a diferença de tempo entre os dois modelos no qual o DBSCAN é exageradamente pior nesta métrica.

3. Conclusões

Neste trabalho foram realizados estudos teóricos e práticos sobre clusterização e dois de seus principais algoritmos o K-Means e DBSCAN. Considerando a parte teórica foram utilizadas referências para estimar as diferenças teóricas entre as técnicas, já considerando a parte prática foram aplicados experimentos na linguagem Rust para a validação dos conceitos teóricos estudados com pré-processamento, aplicação da clusterização e pós-processamento.

Os experimentos validaram para os cenários destas bases de dados que as diferenças teóricas esperadas são válidas na prática, no geral DBSCAN possui a principal vantagem de agrupar os dados em formatos arbitrários enquanto o K-Means possui a principal vantagem de realizar agrupamentos em melhor velocidade, porém em formato elíptico.

Foi perceptível que em casos onde a base possui dados espalhados sem partições de alta densidade como a base Mall Costumers pode ser interessante o uso do K-Means como primeira opção, em cenários com dados particionados em regiões de altas densidades como a base Clustering GMM o uso do DBSCAN deve ser priorizado já em bases com muitos objetos é necessário fazer uma análise, pois embora os dados estejam particionados em regiões densas é preciso levar em consideração o tempo que o DBSCAN leva para realizar o agrupamento onde o K-Means possui melhor velocidade.

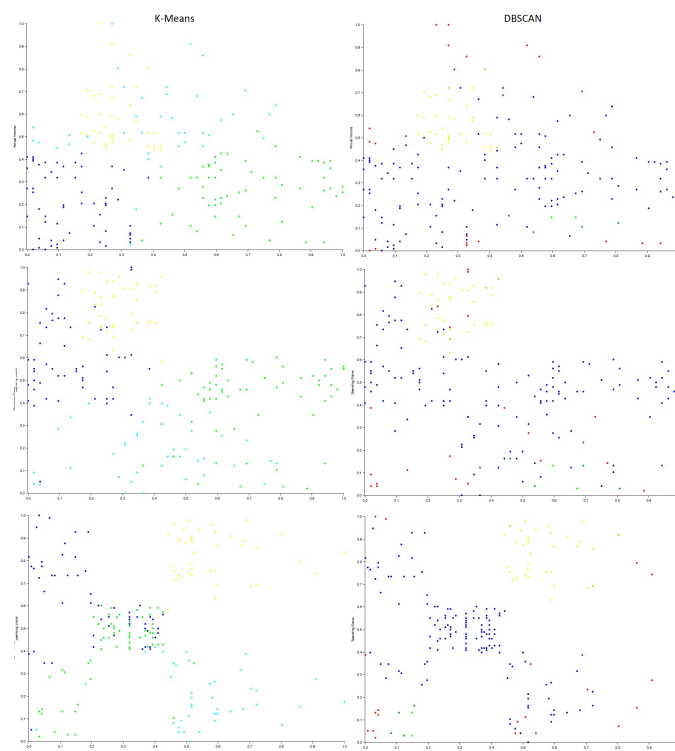


Figure 13. Base de dados Mall Customers

Métricas	K-Means		DBSCAN	
	Coefficiente de Silhueta Global	Tempo Gasto MS	Coefficiente Silhueta Global	Tempo Gasto MS
Íris	0,66	5	0,66	25
Clustering GMM	0,59	10	0,63	79
Mall Costumers	0,39	4	0,14	79
Circular	0,37	19	-0,08	1391
Média	0,5	9,5	0,34	393,5
Total	2,01	38	1,35	1574

Figure 14. Tabela de resultados

4. Trabalhos futuros

Embora foi possível realizar inferências sobre os resultados obtidos as métricas utilizadas foram poucas, fica como um adicional o uso de mais métricas além do coeficiente de silhueta e contagem de tempo.

Os cenários criados foram baseados em hiperparâmetros obtidos de forma experimental, é possível variar os hiperparâmetros para a comparação de mais cenários.

Podem ser usado dados categóricos para a verificação do impacto destes nas visualizações e as análises finais, pois no contexto deste projeto não foram abordadas.

Podem ser empregadas mais técnicas de clusterização para a contraposição juntamente com as duas técnicas empregadas.

O custo do algoritmo DBSCAN com relação a tempo pode ser melhorado.

References

- Bijuraj, L. (2013). Clustering and its applications. In *Proceedings of National Conference on New Horizons in IT-NCNHIT*, volume 169, page 172.
- Ellis, C. (2022a). When to use dbscan.
- Ellis, C. (2022b). When to use k-means clustering.
- Ernerfeldt, E. (2022). Why rust?
- Fairhead, H. (2022). Rust fast and safe.
- Martin Ester, Hans-Peter Kriegel, J. S. X. X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise.
- Mohd, W., Beg, A., Herawan, T., and Rabbi, K. (2012). *MaxD K-Means: A Clustering Algorithm for Auto-generation of Centroids and Distance of Data Points in Clusters*, volume 316, pages 192–199.
- Nvidia (2023). Cluster analysis.
- Sharma, E. (2020). K-means vs. dbscan clustering - for beginners.